UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 9

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título: Estrategia para aplicar Proceso Personal de Software en los proyectos del Polo Productivo de PetroSoft.

Autores: Armando Miguel Fundora Hernández.

Miguel Lantigua Trujillo.

Tutores: Ing. Heydi Menéndez Ávalos.

Ing. Enrique Pérez Rodríguez.

Ciudad de la Habana, 2008.

"Año 50 de la Revolución"

La responsabilidad nuestra es luchar porque la calidad del producto que aquí se haga sea de las mejores y la mejor posible…

Ché

AGRADECIMIENTOS

Mencionándolos no basta para agradecerles todo su apoyo; sin embargo es la única forma de dejar constancia de lo que todos ustedes han hecho por nosotros.

Especialmente agradecemos:

A la revolución que sin ella nada de esto fuera posible.

A nuestras madres Vivian y Josefina, por su apoyo constante en nuestras vidas, y principalmente en los estudios, por todo el amor que hemos recibido de ellas y porque son los seres más maravillosos que existen.

A nuestros padres Miguel y Armando, por lo bien que nos han guiado en la vida, y por lo mucho que hemos aprendido de ellos, dignos ejemplos a seguir.

A nuestros hermanos por el cariño incondicional que nos han brindado.

A mi esposa por el amor que me ha dado y los momentos tan bonitos que he pasado con ella.

A mi hijo Kevin que es mi razón de ser y por la persona que lucho.

A mis abuelos y tíos por ayudarme en mi carrera, y darme mucho amor.

A Barby y familia, por ser otra madre más en mi vida.

A nuestras familias, por confiar y siempre esperar lo mejor de nosotros.

Y para terminar, queremos agradecer al amor y a la vida, que hicieron posible que este día se hiciera

Realidad.

¡Sinceramente, muchas gracias a todos!

Miguel y Armando.

DEDICATORIA

Se lo dedicamos muy en especial a nuestros padres por la gran obra que han venido realizando, a los que nos enseñaron a dar nuestros primeros pasos, a los que lograron hacer de nosotros hombres de bien, a los que ni dedicándoles nuestras vidas recompensaremos jamás.

Miguel y Armando.

DECLARACIÓN DE AUTORÍA

Declaramos ser los autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Armando Miguel Fundora Hernández.

_____

Firma del autor.

Miguel Lantigua Trujillo.

_____

Firma del autor.

Ing. Enrique Pérez Rodríguez.

_____

Firma del tutor.

Ing. Heydi Menéndez Avalos.

_____

Firma del tutor.

DATOS DE CONTACTO

Tutores: Ing. Enrique Pérez Rodríguez.

Ing. Heydi Menéndez Avalos.

Universidad de las Ciencia Informáticas, La Habana, Cuba.

hmenendez@uci.cu.

enriquepr@uci.cu.

OPINIONES DE LOS TUTORES SOBRE EL TRABAJO DE DIPLOMA

Título: Estrategia para aplicar Proceso Personal de Software en los proyectos del polo productivo de PetroSoft.

Tutores: Ing. Enrique Pérez Rodríguez.

Ing. Heydi Menéndez Avalos.

Fecha: Junio 2008

Los tutores del presente Trabajo de Diploma consideramos que durante su ejecución los estudiantes mostraron las cualidades que a continuación se detallan.

Han mostrado independencia. Han sido creativos, mostrando iniciativa en la investigación realizada así como en la gestión de la información y la comunicación con los desarrolladores del polo de PetroSoft. El trabajo se caracteriza por su originalidad. Los estudiantes han mostrado una gran laboriosidad en su trabajo así como una gran responsabilidad.

El trabajo de diploma desarrollado por los estudiantes tiene calidad científica y representa además un aporte significativo para el desarrollo del polo de PetroSoft por cuanto permite una mejora notable en la gestión de los proyectos que integran el mismo a la vez que permiten llevar un seguimiento y control de los datos individuales de sus desarrolladores a través de un PSP que se adecua a las características y necesidades de dicho polo.

Por todo lo anteriormente expresado consideramos que los estudiantes están aptos para ejercer como Ingenieros Informáticos; y propongo que se le otorgue al Trabajo de Diploma la calificación de 5 puntos y sea publicado.

Ing. Enrique Pérez Rodríguez.	Ing. Heydi Menéndez Avalos.
_____	_____
Firma del tutor	Firma del tutor

RESUMEN

La calidad de software adquiere cada día mayor auge e importancia a nivel mundial, las exigencias de los clientes es cada vez más elevada, así como la competitividad entre las empresas. En Cuba se toman las medidas pertinentes para formar parte de este mercado debido a las perspectivas económicas que brinda. De ahí el empeño de las empresas con el objetivo de producir con más calidad, ser más eficientes y con una mayor aceptación en el mercado. El desarrollo de la industria de software implica que los productos deben ser confiables, rápidos de utilizar, precisos y muy bien documentados.

Existen factores que constituyen un elemento crítico y determinante para definir la calidad, como son las pruebas de software, la forma disciplinada y ordenada de estructurar el trabajo personal del ingeniero, constituye una labor determinante a la hora de planificar, medir y gestionar su trabajo, precisamente el uso adecuado del Proceso Personal de Software, contribuye a eliminar el mayor número de faltas existentes en la aplicación y de esta manera obtener el producto final con un mínimo de errores y por consiguiente mayor calidad , por ello el PSP constituye una de las fases más importantes en el desarrollo de un sistema.

Es por ello que la investigación surge debido a que en la Universidad de las Ciencias Informáticas (UCI), no se aplica el Proceso Personal de Software de forma eficiente en los proyectos del Polo Productivo PetroSoft, centrándose la investigación en el como aplicar el Proceso Personal de Software eficientemente, lanzando así una estrategia para darle solución a la misma.

PALABRAS CLAVES

- Calidad de Software.
- Proceso Personal de Software.
- Niveles.
- Fases.
- Etapas.
- Artefactos.
- Datos Históricos.
- Módulos.

ÍNDICE

ÍNDICE DE TABLAS

INTRODUCCIÓN

La informática y dentro de esta, la producción de software han alcanzado en la actualidad, un elevado auge e importancia a nivel mundial. Su desarrollo crece constantemente y con ello la demanda, los productos que se obtienen deben tener la mayor calidad y ser entregados en el tiempo previsto.

En Cuba también se han notado muchos avances en este sentido, no solo por el beneficio que trae para el desarrollo de sistemas sino también con el objetivo de introducirse en el mercado mundial y por supuesto aprovechando sus beneficios económicos. El desarrollo de la industria de software implica que los productos deben ser confiables, rápidos de utilizar, precisos y muy bien documentados. Cuando un producto cumple con estos requisitos se puede decir que se han aplicado las técnicas de Ingeniería de Software correctamente, se han utilizados los roles apropiados y las pruebas hechas al mismo se han aplicado eficientemente para alcanzar el nivel de calidad requerido.

La industria del software ha alcanzado un desarrollo tan elevado que los resultados no cubren las expectativas inicialmente vislumbradas debido a que la productividad que se alcanza, en sentido general es baja, la cantidad de recursos a consumir es alta y por tanto el trabajo realizado casi nunca tiene la calidad requerida.

La calidad de software es otro aspecto que ha adquirido gran importancia hoy en día en el mercado de la industria del software, pues influye de manera directa en la competencia, en el mercado mundial y en la aceptación por parte del cliente. Debido a la importancia que tiene producir con gran calidad, a la evolución y la generación de nuevas técnicas, equipos y procedimientos, se lleva cabo la necesidad de evaluar constantemente los estándares de calidad, tanto de los procesos de producción como del producto final, incorporándose además del aseguramiento de la calidad, aspectos de gestión de la calidad. Cuando se profundiza en las insatisfacciones que afloran de manera reiterada la aplicación inadecuada de las técnicas para el control de la calidad, la no utilización de los roles y procesos apropiados para el desarrollo de las tareas de la empresa de software y la no utilización de modelos de calidad en ellas, se puede llegar a la conclusión de que existen problemas por parte de los ingenieros a la hora de estructurar su trabajo personal, en la evaluación de procesos, en la planificación, medición y gestión de su trabajo.

Precisamente el Proceso Personal de Software (PSP) propone una forma disciplinada de estructurar el trabajo personal, el cual consta de métodos, formularios y procedimientos que ayudan a planificar, medir, y gestionar el trabajo.

El PSP, es una propuesta de evaluación de procesos la cual es determinante para el desarrollo de habilidades y hábitos necesarios para planificar, rutear y analizar proyectos complejos desde una perspectiva personal. PSP plantea siete niveles de evolución a nivel personal, en el nivel PSP0 se utilizan una serie de documentos de trabajo como: Pautas, que sirven de guía para entender los pasos que componen el Proceso, Logs para el registro de datos, Resumen Plan de Proyecto para el informe de los resultados obtenidos en el Proceso. Estos documentos acompañaran al Proceso PSP0, que considera las etapas de Planificación, Desarrollo y Post Mortem. Este Proceso es alimentado por los Requerimientos del Usuario y tiene como salida un Producto terminado, datos del Proyecto y Proceso y el Resumen del Plan de Proyecto. En la etapa de Planificación se produce un Plan para efectuar el trabajo, a continuación se hace el trabajo (Desarrollo) y en la etapa de Post Mortem se comparan los resultados obtenidos con la planificación, se registran los datos y se produce el Resumen del Plan. Como ejemplo, la fase de Desarrollo, en caso de que los requerimientos sean desarrollar un producto de software estaría constituida por las Etapas de Diseño, Codificación, Compilación y Prueba. Siguiendo estos aspectos se considera acertada la construcción del software habiendo aplicado cada una de las etapas del PSP de acuerdo a las necesidades particulares del mismo. [1]

La dirección de calidad de la Universidad de las Ciencias Informáticas se ha trazado como objetivo fomentar el uso de PSP por parte de todos los integrantes de los equipos de desarrollo de los proyectos. La facultad 9, a tono con esta política y comprendiendo la necesidad de la medida, así como las ventajas que la misma entraña, pretende realizar un estudio en cada polo productivo para determinar, de acuerdo a las especificidades de los mismos, cómo aplicar óptimamente PSP por parte de cada uno de los desarrolladores.

Debido a que en los proyectos del Polo Productivo PetroSoft no se aprovecha del todo, se genera la **situación problemática** enunciada a continuación:

En los proyectos del Polo Productivo PetroSoft no existe una estrategia que plantee como aplicar de forma eficiente el PSP, para optimizar el desempeño individual de cada integrante del equipo de desarrollo, lo que trae como consecuencias las siguientes deficiencias:

- Entregas tardías de los productos.
- Una planificación de trabajo deficiente.
- Poco aprovechamiento de los recursos.
- Poca organización de los planes de trabajo.

- Lentitud en el desarrollo de los proyectos.
- Ineficiencia en el desarrollo de los proyectos.

Aspectos que unidos, propician que se corra el riesgo de que el producto final no se entregue con la calidad requerida. Por lo que los realizadores del presente trabajo se plantean el siguiente **problema** a resolver:

¿Cómo aplicar el Proceso Personal de Software de forma eficiente en los proyectos del Polo Productivo PetroSoft?

El **objeto de estudio** de la presente investigación lo constituyen los métodos de ingeniería de desarrollo que plantea el PSP y el **campo de acción**, la estrategia de aplicación del PSP en los proyectos del Polo Productivo PetroSoft.

Para dar solución al problema planteado se traza como **objetivo general**:

Plantear una estrategia que permita aplicar PSP en los proyectos del Polo Productivo PetroSoft.

Y los siguientes **objetivos específicos**:

- Analizar la factibilidad del uso de los niveles de PSP en el desarrollo de software.
- Determinar cuales son los niveles de PSP a aplicar en los proyectos del Polo Productivo PetroSoft.
- Hacer una propuesta de adaptación de métodos por etapa del PSP en los proyectos del Polo Productivo PetroSoft.

Para dar cumplimiento a los objetivos definidos anteriormente se llevarán a cabo las siguientes **tareas de la investigación**:

- Estudiar las características de los proyectos del Polo Productivo PetroSoft, así como del personal que lo integra a través de encuestas y entrevistas.
- Localizar la existencia del problema en el uso de los niveles de PSP en el desarrollo de software.
- Analizar los procedimientos y métodos propuestos por PSP y determinar los ajustes que sea necesario realizar a cada uno de ellos de acuerdo a las especificidades del Polo Productivo PetroSoft.
- Investigar sobre los métodos para la gestión y control de los procesos y actividades del personal del proyecto.
- Investigar sobre el estado del arte sobre la aplicación de PSP en el desarrollo de software.

- Aplicar los métodos propuestos por etapa del PSP a un software en los proyectos del Polo Productivo PetroSoft.

Teniendo en cuenta los elementos expuestos anteriormente se plantea como **idea a defender** para la solución del problema:

Si se cuenta con una estrategia que permita aplicar el PSP de manera eficiente, entonces los proyectos del Polo Productivo PetroSoft tendrán la calidad requerida.

Para la realización de la investigación fue necesario definir la población, la cual la constituyen todos los proyectos del Polo Productivo PetroSoft de la facultad, como unidad de estudio de acuerdo a la naturaleza de la investigación y al diseño teórico elaborado se definió que fueran los métodos de ingeniería de desarrollo que plantea el PSP, utilizadas en los proyectos del Polo Productivo PetroSoft de la facultad, estos son , Conceptualización de Soluciones, el Ministerio de Energía y Petróleo (MEMPET) y el Sistema Automatizado para el Control de la Gestión de Indicadores de las Refinerías (SACGIR) tomando como muestra este último.

Dentro de las diferentes técnicas de muestreo, se utilizó las no probabilísticas, puesto que estas no garantizan que cada elemento de la población se haya incluido en la muestra, pero dentro de ella se utilizó la técnica del muestreo intencional, esta no ofrece resultados confiables, pero se realizó intencionalmente con datos iniciales y comprobaciones externas previas en la muestra seleccionada, por tanto se garantizó que los resultados fueran confiables.

El tamaño de la muestra de la investigación coincide con los integrantes del proyecto del Polo Productivo PetroSoft de la facultad, la muestra en si, por tanto es 30.

El método científico es la estrategia general que orienta y permite organizar globalmente la actividad científica, pero se hace necesaria en cada investigación una estrategia de acuerdo a las condiciones específicas predominantes, atendiendo a la trayectoria del problema y el conocimiento acumulado sobre el mismo, como tipo de estrategia de investigación se definió una investigación explicativa o experimental.

Se definió este tipo de estrategia porque está dirigida a las causas que afectan el fenómeno en estudio, o sea estudiar las causas del uso correcto del PSP en los proyectos del Polo Productivo PetroSoft.

El método científico de la investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad, y el pensamiento con el objetivo de descubrir su esencia y sus relaciones. Dentro de los métodos teóricos y empíricos utilizados se tienen los siguientes:

Métodos teóricos:

Histórico-lógico: En la primera fase de la investigación se desarrolla un estudio del estado del arte de la problemática analizada; revisando de forma crítica cada uno de los documentos para poder resaltar la importancia de las diferentes estrategias de desarrollo de software que se llevan a cabo en la actualidad en el mundo, así como la eficiencia y/o deficiencias de cada una de estas.

Hipotético-deductivo: Permite a partir del problema concreto plantear los objetivos específicos e hipótesis que en el transcurso de la investigación son resueltos siguiendo métodos científicamente bien fundamentados.

Analítico-Sistémico: Plantea el problema como un todo, estudio de metodologías, esquemas del uso de las mismas en el desarrollo de softwares y cada uno de los métodos y herramientas que se utilizan.

Métodos Empíricos:

Experimento: Se usará este método ya que mediante él se verificará la hipótesis y se obtiene conocimiento sobre el objeto de estudio. Se realizarán pruebas para determinar como aplicar el PSP de manera eficiente en los proyectos del Polo Productivo PetroSoft.

Entrevista: La entrevista se aplicará a conocedores del tema, personal capacitado en la muestra, o sea en un proyecto del Polo Productivo PetroSoft en la facultad, para recopilar información sumamente importante para la investigación.

Este documento esta estructurado en tres capítulos:

Capítulo I. Fundamentación Teórica: En el presente capítulo se dará a conocer aspectos importantes sobre la calidad de software, los modelos de calidad del producto, así como las medidas de calidad; la importancia que tiene la garantía de la calidad de un producto, así como su estado en la actualidad en Cuba y en la Universidad de las Ciencias Informáticas, así como un estudio sobre el estado del arte del Proceso Personal de Software , los principales conceptos, métodos , niveles, fases y elementos teóricos de vital importancia para el desarrollo de esta investigación.

Capítulo II. Análisis de la propuesta de solución: En el presente capitulo se realiza un análisis de la situación existente en el proyecto muestra, este consiste en realizar un análisis sobre todos los documentos que genera PSP, plantillas, formularios, tablas. Este análisis se realizará de forma tal que se explique en que consiste el documento , la importancia que tiene hacer el documento y por último se recomienda como mejorarlo, o sea plantear la estrategia de solución, mediante el uso de plantillas, solo las propuestas, con el objetivo de optimizar el uso de PSP y hacerlo menos engorroso.

Capítulo III. Presentación de la propuesta de solución: El presente capítulo pretende realizar una propuesta a la estrategia anteriormente analizada. Para ello se realiza una propuesta de aplicación de PSP por roles, con el objetivo de hacer menos complejo el uso de PSP, y evitar que sea aplicado completamente (todos sus niveles, fases y formularios) por todo el personal, lanzando la propuesta de solución en tres etapas para aplicar PSP.

CAPÍTULO 1

1.1 Introducción.

La industria del software ha alcanzado gran importancia a nivel mundial y con ella la calidad de software, ya que es muy importante asegurar la calidad del producto final, para ello se revisa la documentación asociada al software con el objetivo de verificar su cobertura, corrección, confiabilidad, facilidad de mantenimiento entre otros aspectos relevantes.

El presente capítulo tiene como objetivo exponer los fundamentos teóricos generales que sirven de punto de partida para saber un poco más sobre el Proceso Personal de Software (PSP) porque de esta forma se abordarán aspectos importantes referentes a la calidad de software, así como los modelos de calidad y medidas que pueden ser llevadas a cabo .También se hace alusión a un profundo estudio sobre el Proceso Personal de Software (PSP) y dentro de este se hace énfasis en los niveles , en las medidas para contabilizar el tamaño del software y en los métodos de estimación .

1.2 Calidad de Software.

Uno de los problemas que se han afrontado desde los inicios en la esfera de la computación es la calidad del software. A finales de los sesenta, cada vez que se quería afrontar la mejora del software todo se dirigía a la mejora del código dejando olvidado todo lo que le rodeaba, ya en la década del setenta, este tema fue motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de software, los que fueron mejorando los procesos que se habían definido hasta el momento con el objetivo de mejorar la calidad que seguía siendo aún un tema crítico en el mercado del software, ya en los años ochenta se comenzaron a tener en cuenta los aspectos de especificaciones, diseño, evaluación y gestión del software. Pese a estos cambios realizados, en la actualidad persiste la problemática, ya que el nivel de complejidad del software va aumentando y no se han alcanzado los niveles de calidad deseados.

Para nadie es un secreto que la calidad es sinónimo de eficiencia, flexibilidad, corrección, portabilidad, usabilidad, seguridad, integridad, etc. La calidad del software es medible y varía de un sistema a otro o

de un programa a otro. Es posible medir los principales atributos que forman o caracterizan a un software de buena calidad. La idea es que la calidad del software se caracteriza por ciertos atributos: fiabilidad, flexibilidad, robustez, comprensión, adaptabilidad, modularidad, complejidad, portabilidad, usabilidad, reutilización, eficiencia, y sin lugar a dudas es posible medir cada uno de ellos, y por consiguiente, caracterizar o medir la calidad del software en cuestión. Para cada atributo a medir, hay una medición confiable (objetiva) que puede llevarse a cabo. La idea es que, dado que el atributo deseable es difícil de medir, se mide otro atributo que está correlacionado con el primero, por ejemplo:[2]

- La fiabilidad (software confiable, pocos errores) se mide a través del número de mensajes de error, mientras más mensajes existan, contiene entonces menos errores este software.
- La flexibilidad (capacidad de adaptación a diferentes tipos de uso, a diferentes ambientes de uso) o adaptabilidad.
- La robustez (pocas fallas catastróficas, el sistema “no se cae”) se mide a través de pruebas y uso prolongado.
- La comprensión (capacidad de entender lo que el sistema hace) se mide viendo la cantidad de comentarios que posee el software, y la extensión de sus manuales de usuarios.
- El tamaño se mide en bytes, o sea el espacio que ocupa en memoria, (esta medición no tiene objeción, se mide lo que se quiere medir).
- La eficiencia de un programa se mide en segundos, es la rapidez en su ejecución (esta medición no tiene objeción, se mide lo que se quiere medir).
- La modularidad de un programa se mide contando el número de módulos que lo conforman.
- La complejidad de un programa se mide contando el número de anidaciones en expresiones o postulados, (es lo que se le llama complejidad ciclomática).
- La portabilidad es la facilidad con que se eche a andar en otro sistema operativo distinto al que fue creado. Se mide preguntando a usuarios que han hecho estos trabajos.
- La usabilidad de un programa es alta cuando el programa aporta gran valor agregado a nuestro trabajo. “Es indispensable contar con él”. Se mide viendo qué porcentaje de las necesidades cubre el programa.

- La reutilización de un programa, se mide por la cantidad de veces y que partes del mismo se han reutilizado en otros proyectos de desarrollo de software.
- La facilidad de uso (ergonomía) caracteriza a los programas que no cuesta trabajo aprender, que se amoldan al modo intuitivo de hacer las cosas. Se mide observando la cantidad de pantallas que interaccionan con el usuario, y su sofisticación.

Se concluye que en vez de medir la calidad del producto, se puede medir la calidad del proceso. Tener un buen proceso implica producir software de buena calidad. El problema es que no se sabe cómo deducir cuál proceso producirá buena calidad en el software. En ocasiones se recurre a procesos que suenan o se ven razonables, o que han sido ensayados en otros lados con éxito, o que están dados por algún estándar o comité internacional, sin embargo mientras el proceso que se lleve a cabo sea adaptable a nuestro proyecto y sea totalmente fiable entonces se obtendrá un Software de calidad.

La obtención de un software con calidad implica la utilización de metodologías o procedimientos, estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software. La innovación tecnológica y la adecuación a las nuevas tecnologías plantean el gran reto futuro. Pese a lo arriesgado de hacer predicciones, está claro que la medición se plantea como uno de los principales campos de investigación en Calidad del Software.

La especificación y evaluación integral de la calidad de los productos de software es un factor clave para asegurar que la calidad sea la adecuada. Esto se puede lograr definiendo de manera apropiada las características de calidad, teniendo en cuenta el propósito del uso del producto de software. Es importante especificar y evaluar cada característica relevante de la calidad de los productos de software, mientras esto sea posible, utilizando mediciones validadas o de amplia aceptación, que hagan técnicamente transparente esta actividad.

1.3 Modelos de Calidad del Producto.

A fines de la década del ochenta e inicios de la década del noventa se ha puesto mucho énfasis en los conceptos de calidad del producto y satisfacción del usuario desde diversos enfoques, y particularmente a la valoración y certificación de la calidad de procesos con los bien conocidos Modelo de Madurez de Capacidades Integrado (CMMI) y Software Mejora de Procesos y Capacidad de Determinación (SPICE), sin embargo, también es conocido que los modelos de calidad ya eran reconocidos en la comunidad científica a fines de la década del 70.[3]

Dada la naturaleza lógica del producto, se asume que la calidad de un sistema de software depende de alguna manera de la calidad del proceso usado para desarrollarlo. Los modelos de evaluación y mejora de procesos y su estandarización, han tomado un papel determinante en la identificación, integración, medición y optimización de las buenas prácticas existentes en la organización y desarrollo del software.

Han surgido continuamente con el desarrollo de la informática y las comunicaciones una serie de herramientas, técnicas y modelos que facilitan a las organizaciones encargadas de las tecnologías de la información, generar productos que cumplan las expectativas del cliente e incluso las rebasen, herramientas que prometen ser la solución a los problemas de calidad, costo y tiempos de desarrollo.

Otras normas, directivas, modelos o estándares más usados en la actualidad son básicamente las siguientes:

Familia ISO 9000

ISO 9000 consiste en una serie de procedimientos y directrices que le permiten homogenizar lenguajes y bases técnicas a nivel mundial, con el fin de seleccionar y mejorar procesos. Esta serie de normas pueden aplicarse a cualquier industria, producto o servicio, y consta de requisitos y directrices para establecer sistemas de calidad dentro de una organización, permitiéndole efectuar transacciones con cualquier organización en el mundo, con menor riesgo y mayor confianza, son normas prácticas burocráticas que buscan el logro de la calidad .[3]

Las normas ISO 9000 tiene tres componentes básicos: administración, sistema de calidad y aseguramiento de la calidad.

- Administración.

ISO 9000 provee un sistema para alcanzar el progreso de la organización mediante la realización de metas estratégicas, comprensión de las necesidades de los usuarios, productividad, etc., por medio de acciones correctivas y preventivas.

- Sistema de calidad.

ISO 9000 requiere que la organización documente los procedimientos y los ponga en práctica, de tal forma que si se realiza un cambio, también se registre por escrito. Es necesario contar con una base documental que se ajuste a la realidad al cien por ciento.

- Aseguramiento de la calidad.

ISO 9000 es dinámico, ya que se envuelve en muchas facetas de la organización, como por ejemplo, el establecimiento y documentación de sistemas de ventas, compras, producción, almacenamiento, embarcación e ingeniería, etc.

Objetivos de las normas ISO 9000:[3]

- Establecer sistemas de aseguramiento de la calidad, que garanticen el buen funcionamiento de la empresa y satisfacción de sus clientes.
- Definir el sistema de administración de las actividades que pueden influenciar la calidad de un producto.
- Ayudar a desarrollar un sistema de calidad a nivel mundial, productos de calidad consistente y buena relación con los clientes.

CMM - CMMI

CMM - CMMI es un modelo de calidad del software que clasifica las empresas en niveles de madurez. Estos niveles sirven para conocer la madurez de los procesos que se realizan para producir software.

Niveles CMM - CMMI

Los niveles CMM - CMMI son cinco:

- Inicial o Nivel 1 CMM - CMMI. Este es el nivel en donde están todas las empresas que no tienen procesos. Los presupuestos se disparan, no es posible entregar el proyecto en fechas, se tienen que quedar durante noches y fines de semana para terminar un proyecto. No hay control sobre el estado del proyecto. Si no se sabe el tamaño del proyecto y no se sabe cuanto se lleva hecho, nunca sabrán cuando termina.[4]
- Repetible o Nivel 2 CMM - CMMI. Quiere decir que el éxito de los resultados obtenidos se pueden repetir. La principal diferencia entre este nivel y el anterior es que el proyecto es gestionado y controlado durante el desarrollo del mismo. El desarrollo no es opaco y se puede saber el estado del proyecto en todo momento.[4]

Los procesos que hay que implantar para alcanzar este nivel son:

- Gestión de requisitos.
- Planificación de proyectos.

- Seguimiento y control de proyectos.
- Gestión de proveedores.
- Aseguramiento de la calidad.
- Gestión de la configuración.

➢ Definido o Nivel 3 CMM - CMMI. Resumiéndolo mucho, alcanzar este nivel significa que la forma de desarrollar proyectos (gestión e ingeniería) esta definida, por definida quiere decir que esta establecida, documentada y que existen métricas (obtención de datos objetivos) para la consecución de objetivos.[4]

Los procesos que hay que implantar para alcanzar este nivel son:

- Desarrollo de requisitos.
- Solución Técnica.
- Integración del producto.
- Verificación.
- Validación.
- Desarrollo y mejora de los procesos de la organización.
- Definición de los procesos de la organización.
- Planificación de la formación.
- Gestión de riesgos.
- Análisis y resolución de toma de decisiones.

La mayoría de las empresas que llegan al nivel 3 paran aquí, ya que es un nivel que proporciona muchos beneficios y no ven la necesidad de ir más allá porque tienen cubiertas la mayoría de sus necesidades.

➢ Cuantitativamente Gestionado o Nivel 4 CMM - CMMI. Los proyectos usan objetivos medibles para alcanzar las necesidades de los clientes y de la organización. Se usan métricas para gestionar la organización.[4]

Los procesos que hay que implantar para alcanzar este nivel son:

- Gestión cuantitativa de proyectos.

- Mejora de los procesos de la organización.

- Optimizado o Nivel 5 CMM - CMMI. Los procesos de los proyectos y de la organización están orientados a la mejora de las actividades. Mejoras incrementales e innovadoras de los procesos que mediante métricas son identificadas, evaluadas y puestas en práctica.[4]

Los procesos que hay que implantar para alcanzar este nivel son:

- Innovación organizacional.
- Análisis y resolución de las causas.

Normalmente las empresas que intentan alcanzar los niveles 4 y 5 lo realizan simultáneamente ya que están muy relacionados.

La implantación de un modelo de estas características es un proceso largo y costoso que puede costar varios años de esfuerzo. Aun así el beneficio obtenido para la empresa es mucho mayor que lo invertido.

SPICE

SPICE es un emergente estándar internacional de evaluación y determinación de la capacidad y mejora continua de procesos de ingeniería de software, con la filosofía de desarrollar un conjunto de medidas de capacidad estructuradas para todos los procesos del ciclo de vida y para todos los participantes. Es el resultado de un esfuerzo internacional de trabajo y colaboración y tiene la innovación, en comparación con otros modelos, del proceso paralelo de evaluación empírica del resultado. De acuerdo con la investigación realizada se puede apreciar que la familia ISO y CMMI son los modelos más populares en el mundo de la ingeniería de software, seguido por SPICE.[5]

El modelo describe los procesos que una organización puede ejecutar, adquirir, suplir, desarrollar, operar, evolucionar, brindar soporte de software y todas las prácticas genéricas que caracterizan las potencialidades de estos procesos.

La arquitectura del modelo organiza las prácticas en números de categorías usando diferentes tipos de aproximaciones. La arquitectura distingue entre:[6]

- Prácticas base, son las actividades esenciales de un proceso especifico, agrupado por categorías de procedimientos y procesos de acuerdo al tipo de actividad que direccionan.
- Prácticas genéricas, aplicables a cualquier proceso, que representa las actividades necesarias para administrar el "proceso" y mejorar su potencialidad.

El modelo agrupa a los procesos en cinco categorías:[6]

- Procesos Cliente - Proveedor (Customer - Supplier): esta categoría está dirigida a los procesos que directamente impactan al cliente, al soporte de desarrollo y a la transición del software al cliente.
- Procesos de Ingeniería (Engineering): esta categoría está dirigida a los procesos que directamente especifican, implementan, y mantienen un sistema, un producto de software y la documentación del usuario.
- Procesos de Proyecto (Project): esta categoría consiste en los procesos establecidos dentro del proyecto, coordinación y administración de los recursos para producir un producto o proveer un servicio para satisfacer al cliente.
- Procesos de Soporte (Support): esta categoría consiste en los procedimientos que establecen y soportan el desempeño de los otros procesos del proyecto.
- Procesos de la Organización (Organization): esta categoría consiste en los procesos que establecen las metas del negocio de la organización, los procesos de desarrollo y los recursos que ayudan a la organización a alcanzar dichas metas.

1.4 Medidas de Calidad.

Para elevar la calidad de un producto, la misma debe ser medida de alguna manera. La interrogante principal es la siguiente: ¿qué se mide y cómo se utilizan los datos resultantes? Antes de que se puedan usar los datos, se tiene que decidir sobre las medidas de calidad. Los defectos corregidos en un producto terminado indican la efectividad del proceso de eliminación de defectos. Para manejar la calidad del trabajo, se tiene que medir todo el trabajo, no solamente el producto obtenido. Las medidas de trabajo disponible son: Rendimiento en la eliminación de defectos, Tasa de tiempo por fase, el Índice de Calidad del Proceso (PQI), el Control de la Calidad y Costo de la Calidad (COQ).

1.4.1 Tasa de tiempo por fase.

Otro conjunto de medidas de calidad útiles es la proporción del tiempo utilizado en dos fases del proceso. Por ejemplo, la proporción del tiempo utilizado en la revisión del diseño contra el tiempo utilizado en el diseño. Para medir la calidad del proceso, PSP usa la proporción del diseño contra el tiempo de codificación, revisión del diseño contra tiempo de diseño y revisión de código contra tiempo

de codificación. Se esperaría que el incremento en el tiempo de diseño tienda a incrementar la calidad del producto en correspondencia con la reducción del tiempo de codificación.[7]

Aunque no hay un punto definido sobre cuándo se tiene una calidad pobre, una proporción de 1 a 1 en el diseño contra el tiempo de codificación parece ser un límite inferior razonable, con una proporción de 1.5 como la más óptima. Este punto óptimo varía considerablemente por los individuos y tipos de programas, pero una regla útil es que el tiempo de diseño debe ser, como mínimo, igual al tiempo de codificación. Si esto no sucede, se está haciendo probablemente una significativa cantidad de diseño mientras que se está codificando.[8]

Otra proporción útil es el tiempo de revisión contra el tiempo de desarrollo. En el PSP, la pauta general es que se debe utilizar como mínimo la mitad del tiempo que se usa en la codificación para hacer las revisiones. Esto significa que por cada hora de codificación se debe utilizar como mínimo la mitad de una hora haciendo revisiones del código. La misma proporción sirve para el diseño contra el tiempo de revisión del mismo, requerimientos y tiempo de revisión de estos, etc.[8]

1.4.2 Índice de Calidad del Proceso (PQI).

Para obtener una buena calidad en el proceso, lo primero que se debe hacer es evaluar la calidad de las revisiones de código. Si se sigue el proceso de revisión de código y se usa una lista de chequeo para las revisiones de código se tendrá una revisión con un mayor rendimiento. El mejor indicador de esto es encontrar muy pocos defectos durante la compilación. En el PSP, es considerado como poco, encontrar 10 o menos defectos por KLOC durante la compilación. Sin una buena revisión de código, es bien difícil obtener ese número. Desafortunadamente, si su ambiente de desarrollo no tiene una fase de compilación, estas medidas no estarán disponibles. En este caso, se pueden usar los datos de los tipos de defectos de codificación encontrados en las pruebas de unidad y, en un equipo de TSP, los tipos de defectos de codificación encontrados durante las inspecciones y las pruebas de unidad. [9]

Las métricas en las revisiones de diseño son algo más complejas. Aunque el número de defectos encontrados en las pruebas de unidad puede ser una medida útil de la calidad del diseño, también incluye calidad del código. Si, sin embargo, los defectos por KLOC en la compilación estuvieran por debajo de 10, entonces la calidad es probablemente mejor y los defectos de las pruebas de unidad pueden medir la calidad del diseño. De esta manera, sin los datos de defectos durante la compilación no se puede interpretar las medidas de defectos encontrados durante las pruebas de unidad. Las medidas sugeridas por PSP especifican que si los defectos por KLOC en la fase de compilación son

menos de 10, una cantidad de 5 defectos por KLOC en las pruebas de unidad indican un diseño con calidad. Los criterios de PQI son los siguientes:[8]

1. Diseño/Tiempo de codificación = tiempo de diseño/tiempo de codificación, con rango desde 0.0 a 1.0.
2. Tiempo de revisión del diseño = 2*tiempo de revisión del diseño/tiempo de revisión, con un rango entre 0.0 y 1.0
3. Tiempo de revisión de código = 2*tiempo de revisión de código/tiempo de codificación, con un rango entre 0.0 y 1.0
4. Defectos de compilación/KLOC = 20/(10+defectos de compilación/KLOC), con un rango entre 0.0 y 1.0
5. Defectos de pruebas de unidad/KLOC = 10/(5+defectos de pruebas de unidad/KLOC), con un rango entre 0.0 y 1.0

El índice de calidad del proceso es obtenido multiplicando estos 5 valores entre sí.

1.4.3 Control de la Calidad.

El control de la calidad es una serie de inspecciones, revisiones y pruebas utilizadas a lo largo del proceso del software para asegurar que cada producto cumple con los requisitos que le han sido asignados. El control de la calidad incluye un bucle de realimentación (feedback) del proceso que creó el producto. La combinación de medición y realimentación permite afinar el proceso cuando los productos de trabajo creados fallan al cumplir sus especificaciones. Este enfoque ve el control de la calidad como parte del proceso de fabricación.[8]

Las actividades de control de la calidad pueden ser manuales, completamente automáticas o una combinación de herramientas automáticas e interacción humana. Un concepto clave del control de la calidad es que se hayan definido todos los productos y las especificaciones mensurables en las que se puedan comparar los resultados en cada proceso. El bucle de realimentación es esencial para reducir los defectos producidos.[8]

Se puede ver el proceso de un software como la combinación de dos procesos competitivos: la introducción de defectos y la eliminación. La satisfacción de la eliminación de defectos en los productos terminados viene de la diferencia entre estos dos procesos. Cambios relativamente

pequeños en el rendimiento del proceso pueden causar grandes cambios en la satisfacción con la eliminación de defectos en los productos terminados.

Desde un punto de vista de control del rendimiento, el número final de defectos es el residuo del rendimiento de todas las fases del proceso. Si una fase sencilla para cualquier componente de un producto tiene bajo rendimiento, algún por ciento de defectos que se olvide impactará en la fase subsiguiente y degradará el producto final. Esta degradación también impactará en los costos de mantenimiento, soporte y uso del producto, tanto como el costo de todas las mejoras subsecuentes del producto. Un trabajo de baja calidad tiene un efecto final, por tanto es importante que cada desarrollador maneje la calidad de su trabajo personal. La calidad en el trabajo no solo ahorrará tiempo y esfuerzo, sino que producirá ahorros durante el ciclo de vida del producto.

1.4.4 Costo de la Calidad.

El costo de calidad incluye todos los costos acarreados en la búsqueda de la calidad o en las actividades relacionadas en la obtención de la calidad. Se realizan estudios sobre el costo de calidad para proporcionar una línea base del costo actual de calidad, para identificar oportunidades de reducir este costo, y para proporcionar una base normalizada de comparación. La base normalizada siempre tiene un precio. Una vez que se han normalizado los costos de calidad sobre un precio base, se tienen los datos necesarios para evaluar el lugar en donde hay oportunidades de mejorar los procesos. Es más, se puede evaluar cómo afectan los cambios en términos de dinero.

Los costos de calidad se pueden dividir en costes asociados a la **prevención,** la **evaluación** y los **fallos**.[8]

Entre los costos de **prevención** se incluyen:

- Planificación de la calidad.
- Revisiones técnicas formales.
- Equipos de pruebas.
- Formación.

Entre los costos de **evaluación** se incluyen actividades para tener una visión más profunda de la condición del producto “la primera vez a través de” cada proceso. Algunos ejemplos de costos de evaluación:[8]

- Inspección en el proceso y entre procesos.

- Calibrado y mantenimiento del equipo.
- Pruebas.

Los costos de **fallo** son los costos que desaparecerían sin no surgieran defectos antes del envío de un producto a los clientes. Estos costos se pueden dividir en costos de **fallos internos** y costos de **fallos externos**.

Los costos de **fallos internos** se producen cuando se detecta un error en el producto antes de su envío (retrabajo – revisión, reparación, análisis de las modalidades de fallos.)

Los costos de **fallos externos** son los que se asocian a los defectos encontrados una vez enviado el producto al cliente (resolución de quejas, devolución y sustitución de productos, soporte de línea de ayuda, trabajo de garantía).

Como es de esperar, los costos **relativos** para encontrar y reparar un defecto aumentan dramáticamente a medida que se cambia de prevención a detección y desde el fallo interno al externo.

1.5 La Garantía de la Calidad.

La garantía de la calidad consiste en la auditoría y las funciones de información de la gestión. El objetivo de la garantía de la calidad es proporcionar la gestión para informar los datos necesarios sobre la calidad del producto, por lo que se va adquiriendo una visión más profunda y segura de que la calidad del producto está cumpliendo sus objetivos. Por supuesto, si los datos proporcionados mediante la garantía de la calidad identifican problemas, es responsabilidad de la gestión afrontar los problemas y aplicar los recursos necesarios para resolver aspectos de calidad. [9]

Existe un equipo de Garantía de Calidad el cual es diferente al equipo de desarrollo, especialmente en proyectos grandes. En cuanto al tamaño de este equipo, sirva como indicación que el promedio está en una persona de Garantía de Calidad por cada 15 a 40 personas en el equipo de desarrollo.

Las áreas que caen bajo la responsabilidad del grupo de Garantía de Calidad son tres:[9]

- Las metas y objetivos: Deben asegurar que las metas de la organización en primer lugar, y los objetivos del usuario en segundo lugar se están satisfaciendo, y que no existen conflictos entre ellos, o entre los objetivos de diferentes usuarios.
- Los métodos: Deben asegurar que las actividades de desarrollo de software siguen los procedimientos establecidos, se ajustan a los estándares seleccionados, están de acuerdo

con las políticas de la organización y se ejecutan según las guías de trabajo y recomendaciones disponibles.

- Rendimiento: Debe asegurar que se optimiza la utilización del hardware y software en los productos desarrollados, que son económicos (se desarrollan con el menor costo posible), eficientes (sacan el máximo partido posible a los recursos utilizados) y efectivos (alcanzan el resultado deseado con la menor cantidad posible de recursos, tiempo y esfuerzo).

Las principales tareas del grupo de garantía de calidad, por lo tanto, son:[9]

- Planificación de la calidad: Consiste en seleccionar, clasificar y ponderar las propiedades de calidad que se van a establecer como requisitos, con respecto al producto y con respecto al proceso. Se elegirán también los mecanismos de control de calidad a utilizar para medir y evaluar estas características y se determinarán las metas a alcanzar.
- Supervisión de la calidad: Consiste en supervisar y corregir, si es necesario, el trabajo que se está realizando (según los resultados obtenidos con las actividades de control de calidad), con el objetivo de llegar a satisfacer los requisitos establecidos.
- Construcción de la calidad: Actividades constructivas son aquellas que sirven para "construir" la calidad, es decir, son actividades preventivas cuyo objetivo es evitar la introducción de errores mediante la puesta en práctica de ciertos principios, métodos, formalismos y herramientas.

1.6 Actualidad en la calidad de software en Cuba y en la UCI.

En las últimas dos décadas la industria del software ha emergido, crecido y fortalecido a tal punto que representa actualmente una actividad económica de suma importancia para todos los países del mundo. La industria del software en la mayoría de los países está formada por tejido industrial compuesto en gran parte por desarrolladoras de software que favorecen al crecimiento de las economías nacionales. La mayoría de empresas desarrolladoras de software son pequeñas (tienen menos de 50 empleados) y desarrollan productos significativos que, para su construcción, necesitan prácticas eficientes de Ingeniería de Software adaptadas a su tamaño y tipo de negocio.

La Industria del Software actual afronta una crisis debido a una serie de factores entre los que se encuentran: insuficiente calidad del producto final, estimaciones de duración de proyectos y asignación de recursos inexactas, retrasos en la entrega de productos terminados, están fuera de control los costos de desarrollo y mantenimiento de productos, escasez de personal calificado en un mercado laboral de alta demanda, y una tendencia al crecimiento del volumen y complejidad de los productos.

Las piedras angulares del proceso de mejora continuo del desarrollo del software son: el personal, el proceso y la tecnología. La calidad del software se puede definir como el “Grado con el cual el cliente o usuario percibe que el software satisface sus expectativas”.

La garantía de la calidad del software es una “actividad de protección” que se aplica a cada paso del proceso de software.

Para llevar a cabo adecuadamente una garantía de calidad del software, se deben recopilar, evaluar y distribuir todos los datos relacionados con el proceso de ingeniería del software. La capacidad de garantizar la calidad es la medida de madurez de la disciplina de ingeniería.

Con el objetivo de lograr la informatización de la sociedad cubana, en el país se está trabajando fehacientemente en el desarrollo de la Industria Cubana del Software, la cual permite proveer de sistemas de información no sólo en beneficio de la sociedad cubana, sino también que se pueda exportar productos y que sean reconocidos internacionalmente, sin embargo las empresas que producen software hoy requieren aún de un sistema que cree un entorno organizado donde sean aplicados procedimientos de ingeniería de software que guíen el control de los procesos para desarrollar y mantener el software con una total calidad, para lograr evolucionar hacia una cultura de ingeniería de software y de administración de excelencia.

Dentro de esta esfera de la producción de software se enmarca la Universidad de las Ciencias Informáticas, la cual fue creada con dos objetivos básicos, la informatización de la sociedad y la exportación. Países como Venezuela se ha beneficiado ya con los logros obtenidos, se puede plantear entonces que aún se están limando detalles en lo que a calidad de software se refiere.

La Universidad de las Ciencias Informáticas está envuelta en proyectos de producción de software, por lo que es lógico deducir la necesidad de llevar un control de la calidad de los mismos. Existe un Grupo de Calidad encargado del control de la calidad de los productos de software de todas las facultades. Durante el desarrollo de los primeros proyectos, dicho grupo comenzó a realizar pruebas a los productos, y se determinaron muchos errores en las aplicaciones; es por eso que se hacia necesario la búsqueda de una solución a ese problema. Por tal motivo se decide incorporar al Grupo de Calidad un mayor número de personas involucradas en el proceso de control de la calidad. Por cada facultad se selecciona un Asesor de la Calidad, preferiblemente con conocimientos de Ingeniería de Software, y un grupo de 30 estudiantes para llevar a cabo las actividades de calidad. Estos grupos tributaban tanto a la producción de las facultades, como a nivel de universidad.

Inicialmente se contaba con un grupo de 30 estudiantes de toda la UCI, que trabajaban en todos los proyectos, después es que surgen los Asesores de Calidad por facultades. A partir de este momento se comienzan a desarrollar cursos de adiestramiento para los estudiantes y Asesores involucrados en la calidad. Se establecieron además los Lineamientos mínimos de Calidad en la universidad. Actualmente en la universidad el Grupo de Calidad lleva a cabo las actividades de Pruebas de Caja Negra basadas en la búsqueda del camino básico, para el control de la calidad de los productos de software. Se realizan pruebas de aceptación, integración y funcionalidad.

En la Universidad de las Ciencias Informáticas (UCI) se creó un grupo de trabajo que es el encargado de garantizar la calidad de los productos de software que se están desarrollando, de manera que se puede apreciar que un tema tan importante como la calidad no se ha dejado atrás y de esta forma se han logrado grandes aportes por parte de los estudiantes y profesores.

1.7 Pruebas de Software.

El proceso de pruebas de software es un elemento crítico para la garantía de la calidad del software. Es una actividad en la cual un sistema o unos de sus componentes se ejecutan en circunstancias previamente especificadas e integrada durante todo el ciclo de vida. Es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. La prueba no puede asegurar la ausencia de defectos, solo puede demostrar que existen defectos en el software[10]. La prueba de software es un conjunto de herramientas, técnicas y métodos que hacen a la excelencia el desempeño de un programa. Es también la mejor publicidad que una empresa dedicada a la producción de software pueda tener. Involucra las operaciones del sistema bajo condiciones controladas y evaluando los resultados. Probar es un proceso de ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Dentro del proceso de pruebas, las técnicas para encontrar problemas en un programa son extensamente variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo en tiempo de esta actividad[11]. Las pruebas constituyen un método más para poder verificar y validar el software. Se puede definir la verificación como[12] el proceso de evaluación de un sistema o de uno de sus componentes para determinar si los productos de una fase dada satisfacen las condiciones impuestas al comienzo de dicha fase, la validación como el proceso de evaluación del software al final del proceso de desarrollo para asegurar el cumplimiento de las necesidades del cliente. El objetivo de la etapa de pruebas de manera general es garantizar la calidad del producto desarrollado, y para lograr esto se hace necesario[13]:

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias solo al final de la iteración.
- Diseñar e implementar las pruebas creando los casos de prueba que especifican que probar, creando componentes de pruebas ejecutables para automatizar las pruebas.
- Realizar las diferentes pruebas y manejar los resultados de cada una de forma sistemáticas. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados.

La aplicación de pruebas comienza por lo pequeño y progresa hacia lo grande. Por esta razón, se deben comenzar las primeras pruebas sobre el componente elemental y aplicar:

1.7.1 Pruebas de Unidad.

Es la escala más pequeña de la prueba, está basada en la funcionalidad de los módulos del programa, como funciones, procedimientos, módulos de clase, etc. En ciertos sistemas también se verifican o se prueban los drivers y el diseño de la arquitectura. Para probar los componentes implementados como unidades individuales, se realizan las pruebas de especificación o de caja negra, que verifica el comportamiento de la unidad observable externamente y pruebas de estructura, o de caja blanca, que verifica la implementación interna de la unidad[14].

1.7.2 Pruebas de Caja Negra.

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose solamente en los requisitos funcionales del sistema. Muchos autores consideran que estas pruebas permiten encontrar[14]:

- Funciones incorrectas o ausente.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.

- Errores de inicialización y terminación.

Para desarrollar la prueba de Caja Negra existen varias técnicas, entre ella están:

- Técnica de la Partición de Equivalencia: Es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de casos de prueba que hay que desarrollar.
- Técnica del Análisis de Valores Límites: Esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de Grafos de Causa-Efecto: Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones. Consiste en crear un grafo causa/efecto a partir de las especificaciones, y seleccionar suficientes casos de pruebas para asegurar la cobertura del grafo, y mediante él, construir una tabla de decisión para reflejar las dependencias de los resultados.
- Técnica de Adivinación de errores: Esta técnica consiste en tratar de imaginar cuáles son los errores que se pueden haber cometido con mayor probabilidad, y generar casos de prueba para comprobar dichos errores.

El aspecto humano es esencial en la prueba de Caja Negra aplicando factibles sucesos de la vida real a la prueba, errores de escritura, trabajar en aplicaciones equivocadas creyendo trabajar en la aplicación deseada.

1.7.3 Pruebas de Caja Blanca.

También llamadas estructurales o de cobertura lógica. En ellas se pretende indagar sobre la estructura interna del código, omitiendo detalles referidos a datos de entrada o salida. Su objetivo principal es probar la lógica del programa desde el punto de vista algorítmico. Para esta prueba se consideran tres importantes puntos:[15]

- Conocer el desarrollo interno del programa, determinante en el análisis de coherencia y consistencia del código.
- Considerar las reglas predefinidas por cada algoritmo.

- Comparar el desarrollo del programa en su código con la documentación pertinente. La primera parte de esta prueba es el análisis estático.

Una técnica de prueba de Caja Blanca es la prueba del camino básico, los pasos que se siguen para aplicar esta técnica son:

- A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo.
- Se determina un conjunto básico de caminos independientes.
- Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

1.8 Introducción al Proceso Personal de Software (PSP).

El Proceso Personal de Software (PSP) es un conjunto estructurado de descripciones de procesos, mediciones y métodos que pueden ayudar a que los ingenieros mejoren su rendimiento personal. Proporcionan las formas, guiones y estándares que les ayudan a estimar y planificar su trabajo. Muestra cómo definir procesos y cómo medir su calidad y su productividad. Un principio PSP fundamental es que todo el mundo es diferente y que un método que sea efectivo para un ingeniero puede que no sea adecuado para otro. Así pues, el PSP ayuda a que los ingenieros midan y sigan la pista de su trabajo para que puedan encontrar los métodos que sean mejores para ellos.

PSP mantiene los siguientes principios:[3]

- La calidad de un sistema de software está determinada por la calidad de sus peores componentes.
- La calidad de un componente de software está gobernada por el individuo que lo desarrolla.
- La calidad de un componente de software está gobernada por la calidad de los procesos usados para desarrollarlo.
- La clave para la calidad son las habilidades individuales del desarrollador, compromiso y disciplina personal del proceso.
- Como un profesional del software, es responsable de su proceso personal.

- Medir, monitorear y analizar su trabajo.
- Aprender variaciones de desempeño.
- Incorporar lecciones aprendidas en prácticas personales.

Una organización disciplinada de ingeniería de software tiene bien definidas sus prácticas. Sus profesionales emplean dichas prácticas, monitorean y se esfuerzan para mejorar el desempeño personal y mantienen una responsabilidad por la calidad de los productos que producen. Lo más importante es que poseen los datos y confianza, necesarias para resistir demandas de acuerdos irrazonables.

El único propósito de PSP es ayudar a mejorar las habilidades de ingeniería de software. El mismo constituye una herramienta poderosa que puede ser empleada de muchas maneras. Por ejemplo, lo ayudará a administrar el trabajo, evaluar sus talentos y formar sus habilidades. Contribuye a la realización de mejores planes, para monitorear el desempeño de forma precisa y medir la calidad de los productos. Independientemente de que se diseñe programas, se desarrolle requerimientos, se escriba la documentación o se de mantenimiento al software existente, el PSP ayuda a hacer mejor el trabajo.[3]

En lugar de usar un enfoque para cada trabajo, es necesario un arreglo de herramientas y métodos así como las habilidades expertas para usarlas apropiadamente. El PSP brinda los datos y técnicas de análisis necesarios para determinar que tecnologías y métodos trabajan mejor.

El PSP también brinda un marco de trabajo para comprender, cometer errores y cómo es mejor encontrarlos, arreglarlos y prevenirlos. Es posible determinar la calidad de las revisiones, los tipos de defectos que comúnmente se obvian y los métodos de calidad que son más efectivos. Después de conocer PSP será capaz de decidir qué métodos emplear para usar y cuándo usarlos. También conocerá cómo definir, medir y analizar su propio proceso. Entonces, a medida que se obtenga experiencia, se reforzarán sus procesos para tomar ventaja de cualquier nueva herramienta y métodos de desarrollo.

El PSP no es la respuesta mágica a todos los problemas de ingeniería de software, pero ayuda a identificar dónde y cómo puede mejorar.

1.8.1 Niveles de PSP.

Los siete niveles de procesos empleados para introducir PSP se muestran a continuación. Cada nivel se conforma en el nivel anterior mediante la incorporación de algunos pasos de procesos para el

mismo. Esto minimiza el impacto de los cambios de procesos en el ingeniero, quién necesita solamente adaptar las nuevas técnicas en una línea base de prácticas existentes. Está formado por diferentes componentes entre los que se encuentran:[3]

Guiones: Documentan el Proceso. Contienen los criterios de entrada de cada etapa del proceso, las etapas/pasos y los criterios de salida. Su propósito es guiar al ingeniero para usar el proceso.

Mediciones: Miden el proceso y el producto. Proveen la visibilidad de cómo está trabajando el proceso y del estado actual del trabajo.

Formas: Proveen un medio consistente y conveniente para recolectar y almacenar datos.

Estándares: Proveen definiciones consistentes que guían el trabajo y la recolección de datos.

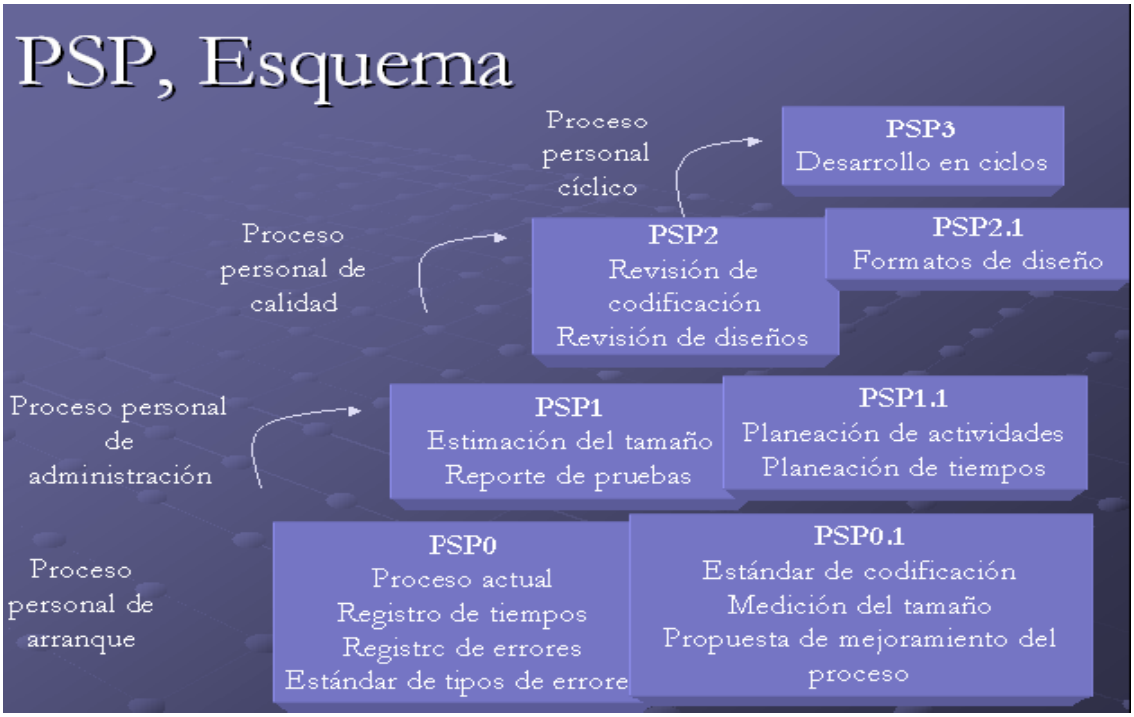

Figura 1: Los niveles de procesos de PSP.

1.8.2 La Línea Base del Proceso Personal - PSP0 y PSP0.1

La Línea Base del Proceso Personal (PSP0 y PSP0.1) brinda una introducción al PSP y establece una base inicial de tamaños históricos, tiempo y datos de defecto. Los ingenieros escriben programas en este nivel. Está permitido que empleen sus métodos actuales pero dentro del marco de trabajo de los 6 pasos de la línea base de procesos que se muestra en la siguiente tabla:

Paso	Fase	Descripción
1	Planificación	Se planifica el trabajo y se documenta el plan.
2	Diseño	Se diseña el programa.
3	Codificación	Se implementa el diseño.
4	Compilación	Se compila el programa, se repara y registran los defectos encontrados.
5	Prueba	Se prueba el programa, se repara y registran los defectos encontrados.
6	Postmortem	Se registra el tiempo actual, defectos y tamaño de datos en el plan.

Tabla 1. Fases del Proceso Personal de Software.

PSP0 introduce procesos básicos de medición y planificación. El tiempo de desarrollo, defectos y el tamaño de los programas son medidos y registrados en formularios establecidos. Un formulario simple de resumen del plan es usado para documentar los resultados planificados y actuales. Un formulario para registrar propósitos de mejora de procesos (PIP) es también introducido (PSP0.1). Dicho formulario permite a los ingenieros contar con una vía conveniente para registrar los problemas del proceso y las soluciones propuestas.

El principal objetivo de PSP0, el proceso de línea base, es proporcionar un marco de trabajo para escribir su primer programa PSP y para obtener datos en su trabajo. Los datos de los primeros programas PSP brindan una línea base comparativa para determinar el impacto de los métodos PSP en su trabajo.

El proceso PSP0 proporciona los siguientes beneficios:[16]

- Una estructura conveniente para ejecutar tareas a pequeña escala.
- Un marco de trabajo para la medición de esas tareas.
- Los fundamentos para la mejora de proceso.

Una estructura de trabajo conveniente:

Con un proceso definido como PSP0 es posible disminuir el tiempo empleado para la planificación de tareas, debido a que ya se ha determinado cómo hacer las tareas cuando el proceso fue desarrollado. Cuando se ejecutan tareas similares muchas veces, es más eficiente desarrollar y documentar el

proceso una sola vez, en lugar de parar para inventar un nuevo proceso cada vez que se inicie un trabajo.

Un marco de trabajo de medición:

Un proceso definido establece las medidas del proceso. Esto permite adquirir datos sobre el tiempo que se emplea en cada tarea, los tamaños de los productos que se elaboran y el número de defectos que son insertados y eliminados en cada paso del proceso. Esos datos ayudan a analizar el proceso, para comprender sus fortalezas, debilidades y mejorarlo. Un proceso definido también brinda mediciones con un significado explícito. Con la línea base del proceso PSP0 algunas de las entradas y criterios de salida no son muy precisos. Esto es porque PSP0 es un proceso simple.

Una fundación para la mejora:

Si no se conoce lo que se está haciendo, es difícil mejorar la forma de hacerlo. La mayoría de los profesionales del software defenderían que conocen lo que hacen cuando desarrollan software. Sin embargo, raramente pueden describir sus acciones en detalles. Cuando se sigue un proceso definido, se tiene mayor conocimiento del trabajo que se hace. Es posible observar el comportamiento propio y a menudo se puede observar cómo descomponer un gran paso en elementos más pequeños. PSP0 es un primer paso en la construcción de esta idea más refinada.

1.8.3 La Gestión de Proyectos Personal – PSP1 y PSP1.1

PSP1 y PSP1.1 se enfocan en técnicas personales de gestión de proyectos, introduciendo tamaño y estimación de esfuerzo, planificación de cronogramas y métodos de seguimiento de cronogramas. Los estimados de tamaño y esfuerzo son realizados mediante el método PROBE, Estimación Basada en Proxy. Con este método los ingenieros usan el tamaño relativo de un proxy para hacer sus estimados iniciales, entonces usan datos históricos para convertir el tamaño relativo del proxy a Líneas de Código (LOC). Ejemplos de proxy para la estimación del tamaño del programa son: objetos, funciones y procedimientos. Otros ejemplos incluyen objetos de pantallas, guiones, reportes y documentos impresos. [3]

PSP usa el método de Valor Ganado para la planificación de cronogramas y monitoreo. Dicho método es una técnica estándar de gestión que asigna un valor planificado a cada tarea en un proyecto. Dicho valor está basado en el por ciento del esfuerzo total planificado para el proyecto que dicha tarea tomará. A medida que las tareas se completen, el valor planificado de las tareas se convierte en valor ganado para el proyecto. El valor ganado del proyecto se convierte entonces en un indicador del por

ciento del trabajo completado. Cuando se monitorea semana a semana, el valor ganado del proyecto puede ser comparado con sus valores planificados para determinar el estado, estimar tasas de progreso y proyectar la fecha de culminación del proyecto. [3]

Uno de los escalones de la estructura incremental de PSP es específicamente PSP1, el cual introduce la Estimación de Tamaño y el Reporte de Prueba. El objetivo de PSP1 es entablar un procedimiento ordenado y repetido para la estimación de tamaño de software y como los demás niveles de PSP viene acompañado por un conjunto de guiones y formularios.

Los objetivos de PSP 1.1 tienen bastante similitud con los vistos en PSP 1, sin embargo se introducen otros métodos de planeación y los formatos que se han venido utilizando sufren de pequeñas modificaciones.

En el PSP 1.1 se introducen dos nuevas formas de planeación, el Plan de Tareas y el Plan de Cronograma. El primer tipo se basa en la actividad a desarrollar, como escribir un programa o un reporte. El segundo tipo está basado en un periodo determinado de tiempo, como ejemplo se puede tomar cualquier segmento de un calendario (días, semanas, meses o años). Estos planes nos proporcionan un seguimiento del progreso mientras se está trabajando. Ambos tipos dependen uno del otro para que se lleven a cabo.

En PSP 1.1 el primer paso para hacer una planeación de producto es tener una definición clara del producto. Es necesario tomar en cuenta tres puntos importantes: [3]

- El tamaño y las características importantes del producto.
- Un estimado del tiempo requerido para realizar el proyecto.
- Un calendario del proyecto.

Mientras más complejos sean los productos, éstos requerirán de una planeación más sofisticada. Los planes individuales de producción colaboran a cumplir con las fechas y tareas independientemente, con lo que puede revisar los compromisos adquiridos constantemente.

Los ingenieros deben utilizar planes de producción para saber el estado en el que el proyecto se encuentra, si el proyecto va cumpliendo con el calendario, entonces los programadores puntuales pueden prestar su ayuda a los demás integrantes del equipo. Esta es la manera en la que pueden organizar su tiempo y evitar crisis de último minuto, derivando así, productos de mejor calidad.

Así mismo se requieren de muchos tipos de información tales como acuerdos o asignaciones de responsabilidades, planes de apoyo, especificaciones del producto o del proceso, dependencias de

otros grupos o pruebas especiales. Es importante realizar un plan que sea apropiado a la magnitud y complejidad del trabajo que se realizará. Por medio de la comparación de datos en proyectos pasados, se podrá predecir el tiempo aproximado que llevará realizar el presente proyecto.

PSP 1.1 hace notar que para realizar planeaciones será necesario definir los siguientes términos:[3]

- Producto- Es algo que se produce junto con un colaborador, proveedor o un cliente.
- Proyecto- Un proyecto produce siempre un producto. Es algo que se planea.
- Tarea- Es un elemento definido de un trabajo.
- Proceso- Define la manera de realizar proyectos.
- Planes- Describe la forma de cómo un proyecto específico se debe realizar, ¿Cómo?- ¿Cuándo?- ¿A qué precio?
- Trabajo- Es algo que se lleva a cabo, sea un proyecto o una tarea.

1.8.4 La Gestión Personal de la Calidad – PSP2 y PSP2.1

PSP2 y PSP2.1 incorporan métodos de gestión de calidad para el PSP: diseño personal y revisiones de código, una notación para el diseño, plantillas de diseño, técnicas de verificación del diseño y medidas para la administración de procesos y calidad de producto.

La meta de la gestión de la calidad en PSP es encontrar y remover todos los defectos antes de la primera compilación. La medida asociada con esta meta es el rendimiento. El rendimiento se define como el por ciento de los defectos insertados antes de compilar que fueron eliminados después de compilar. Un rendimiento de un 100% ocurre cuando todos los defectos insertados antes de compilar son eliminados después de compilar. [8]

Dos nuevos pasos del proceso, la revisión del diseño y la revisión del código, están incluidas en PSP2 para ayudar a los ingenieros a alcanzar un 100% de rendimiento. Las mismas constituyen revisiones personales conducidas por un ingeniero en su propio diseño o código. Son revisiones estructuradas y orientadas a los datos que se guían por listas personales de chequeo para la revisión derivadas de los datos históricos de defectos de los ingenieros.

Comenzando con PSP2, los ingenieros también comenzarán usando datos históricos para planificar la calidad y el control de calidad durante su desarrollo. La meta de los ingenieros es remover todos los defectos insertados antes de la primera compilación. Durante la planificación, estiman el número de defectos que serán insertados y eliminados en cada fase. Entonces se usa la correlación histórica

entre tasas de revisión y el rendimiento para planificar revisiones efectivas y eficientes. Durante el desarrollo se controla la calidad mediante el monitoreo de los defectos actuales insertados y eliminados contra el plan, así como mediante la comparación de las tasas actuales de revisión para establecer límites.

Las revisiones son muy efectivas para la eliminación de la mayoría de los defectos encontrados en la compilación y muchos de los defectos encontrados en las pruebas. Pero para sustancialmente reducir los defectos de prueba, se necesitan mejores diseños de calidad. PSP2.1 cubre esta necesidad mediante la incorporación de una notación de diseño, cuatro plantillas de diseño y métodos de verificación de diseño para el PSP. El propósito es no introducir un nuevo método de diseño, pero hay que asegurar que el diseñador examine y documente el diseño desde diferentes perspectivas. Esto mejora el proceso de diseño y hace más efectiva la verificación y la revisión del diseño. Las plantillas de diseño en PSP brindan 4 perspectivas en el diseño: una especificación operacional, una especificación funcional, una especificación de estado y una especificación lógica.

En el nivel 2 de PSP se introduce la gestión de la calidad como uno de los principales aspecto que debe estar presente en cada etapa del desarrollo de un proyecto.

La definición de calidad está centrada en las necesidades de los usuarios y su conformidad con los requerimientos. Es por esto que la idea central, cuando se construye un software, debe partir de: ¿quiénes son los usuarios?, ¿qué es importante para ellos?, ¿cómo se relacionan sus prioridades con respecto a la forma en que construimos, agrupamos o le damos soporte al software?

Si un producto tiene muchos defectos que no garantizan una ejecución consistente, el usuario no apreciará el resto de las cualidades o atributos que este pueda poseer. Esto no significa que los defectos son siempre la máxima prioridad pero si que son una parte muy importante en el desarrollo: si el mínimo nivel de calidad no se logra, más nada es importante.

Por la importancia de un software sin defectos, las organizaciones consagran la mayor parte de su tiempo a encontrarlos y corregirlos y por el hecho de que la magnitud de este proceso de corrección con frecuencia es subvalorada, la mayoría de los proyectos se preocupan por la corrección del defecto e ignoran otras preocupaciones importantes de los usuarios. Cuando un equipo de proyecto está inmerso en la tarea de arreglar los defectos en la fase de pruebas, generalmente ya está escaso de tiempo y las presiones para la entrega son tan intensas que se olvidan otras preocupaciones. Sin embargo, la corrección de estos defectos críticos solo garantiza que el producto alcance el mínimo nivel de calidad deseado. En ese momento: ¿Qué se ha hecho para hacer el producto usable e

instalable? ¿Qué hay de la compatibilidad, ejecución, seguridad? ¿Alguien ha revisado la documentación y si se ha logrado un diseño conveniente para una futura mejora del producto? Por el excesivo tiempo de corrección de los defectos no se ha dedicado ni siquiera un mínimo de tiempo a pensar en los problemas que últimamente son más importantes para los usuarios, sin embargo, con pocas pruebas de defectos, los proyectos pueden dirigirse a los aspectos de calidad que el usuario siente que son más importantes.

Los defectos no son la prioridad máxima, pero un manejo adecuado de los mismos es esencial para el manejo de costo, planificación y otros aspectos de la calidad de un producto. Los defectos son el resultado de errores individuales, por tanto para manejarlos se debe manejar el comportamiento personal. Somos la fuente de los errores en los productos por tanto somos los únicos que podemos prevenirlos y los más indicados para encontrarlos y corregirlos.

PSP 2 tiene como objetivos introducir revisiones de diseño, de código y métodos para evaluar y mejorar la calidad de las revisiones individuales. Aunque la detección y corrección de defectos es un punto de crítica importancia, esto tiene una estrategia defensiva inherente. Para los niveles de calidad requeridos en la actualidad, se tienen que identificar las causas de los defectos, definir pasos para prevenirlos y así se ganará en tiempo y en productividad.

EL proceso PSP toma ventaja del hecho de que los errores de las personas son predecibles. Esto no quiere decir que sean incompetentes, sino que son humanos y como humanos se tiende a repetir los errores. Los principios de los procesos de revisiones personales son los siguientes:[8]

- Revise personalmente todo el trabajo propio antes de moverse a la próxima fase de desarrollo.
- Esfuércese para corregir todos los defectos antes de darle el producto a alguien más.
- Use una lista de chequeo personal y siga un proceso de revisión estructurado.
- Siga las prácticas probadas de revisión: revise en pequeños incrementos, haga revisiones en papel y hágalo cuando está fresco y descansado.
- Mida el tiempo de revisión, el tamaño de los productos revisados y el número y tipos de defectos encontrados. Use datos para mejorar el proceso personal de revisión.
- Diseñe e implemente sus productos de manera que sean fáciles de revisar.
- Revise sus datos para identificar formas de prevenir los defectos.

1.8.5 Proceso Personal Cíclico – PSP3

Desarrollo cíclico**:**

Se le llama desarrollo cíclico cuando se desarrolla una y otra vez. Desarrollos que se repiten periódicamente. Este desarrollo en ciclo actualmente es muy común verlo ya que con el desarrollo de las veces anteriores nos sirve de base para los desarrollos posteriores.

¿Qué es PSP3?

Con PSP3 finalmente el último nivel de PSP es alcanzado. Con este nuevo nivel se introduce una nueva fase, la fase de realizar el proceso personal creado de una manera cíclica y uniforme. Esto quiere decir que a estas alturas del proceso, el programador tiene una manera de programar única y bien definida, es la firma que cada programador debe de poseer.

Obviamente el proceso personal que el programador crea, es un proceso eficaz y aplicable a cada programa que quiera desarrollar. Este nivel ayuda al desarrollador a desarrollar programas más largos en poco tiempo y con menos errores. Es un ejemplo del proceso personal a larga escala porque puede ampliar los métodos del PSP a proyectos mayores, muy adecuado para programas con mas de 1000 LOC.

El Proceso Personal Cíclico, PSP3, centra las necesidades para escalar eficientemente el PSP para proyectos de mayor tamaño sin sacrificar calidad o productividad. En la clase los ingenieros aprenden que su productividad es mayor entre un rango mínimo y máximo de tamaño. Por debajo de este rango la productividad declina debido a los costos fijos superiores. Por encima de este rango, la productividad declina porque el límite de escalabilidad del proceso ha sido alcanzado. PSP3 localiza dicho límite de escalabilidad mediante la introducción de una estrategia cíclica de desarrollo donde grandes programas son descompuestos en partes para el desarrollo y entonces se integran. Esta estrategia asegura que los ingenieros están trabajando a su máxima productividad y los niveles de calidad del producto aumentan, de forma incremental, no exponencial, para grandes proyectos.[1]

Para soportar este enfoque de desarrollo PSP3 introduce el diseño de alto nivel, la revisión de diseño de alto nivel, la planificación del ciclo y ciclos de desarrollo basados en los procesos PSP2.1. Dos nuevos formularios son también introducidos: un resumen del ciclo para resumir tamaño, tiempo de desarrollo y defectos para cada ciclo; y un registro de seguimiento de aspectos para documentar aspectos que pueden afectar ciclos futuros o ciclos completados. Usando PSP3, los ingenieros descomponen su proyecto en una serie de ciclos PSP2.1, entonces integran y prueban la salida de

cada ciclo. Debido a que los programas que se producen con PSP2.1 son de alta calidad, los costos de integración y prueba son minimizados. [2]

Estrategia que se sigue con PSP3.

En la tendencia cíclica de PSP3, el programa se divide en segmentos que se solucionan con PSP2.1. El producto de un ciclo sirve de base para el próximo ciclo.

Los objetivos de PSP 3 son los mismos que se detallan en PSP2.1, las únicas adiciones que se hacen es que el programador debe de ser capaz de desarrollar programas de hasta miles de LOC y para esto se introducen nuevos guiones, formularios y plantillas que son los procesos finales que el programador debe de dominar.

Este proceso comienza con los requerimientos, luego se produce un diseño conceptual del programa entero donde se estima el tamaño y se planifica el desarrollo.

A partir de este diseño de alto nivel, se divide el producto y se divisa la estrategia cíclica. Cada ciclo se ejecuta con PSP2.1: se codifica, se compila, se revisa y se prueba. Depende de cada programador realizar la división de los módulos y la correcta integración de éstos al producto final. Después de cada ciclo hay que reevaluar el plan, elaborar un reporte de ciclo indicando problemas y desviaciones.

1.9 Medidas para contabilizar el tamaño de un Software.

Existen varias formas de medir el tamaño del software, la cual debe ser útil para la estimación del tiempo de desarrollo por esto esta medición debe ser precisa, específica y automáticamente contable.

Precisa: da el valor exacto para el tamaño de un producto.

Específica: está basada en las propiedades definidas de un producto.

Contable: porque para grandes programas es impracticable el conteo manual.

La teoría de Humphrey se basa en la estimación por criterio de expertos, o sea, donde se parte de una experiencia en el desarrollo de un determinado proceso y se utiliza para realizar una estimación de un nuevo proceso.[2]

Las medidas de tamaño se introducen para ayudar a estimar el tamaño del producto. En muchas ocasiones se utiliza como medidas de tamaño las líneas de código (LOC) de un programa pero en la práctica existen otras unidades que pueden ser utilizadas. A continuación se presentan ejemplos clásicos para contabilizar el tamaño de un software:[2]

Estableciendo el conteo estándar de las BD

Producir una BD contar campos y tablas.

Producir programas para usar una BD, contar sentencias, elementos, tablas o líneas de código.

Usar una herramienta GUI contar botones, cajas de dialogo, labels, etc.

Estableciendo un conteo de estándar de líneas de código

Las líneas de códigos (LOC) es una medida útil porque se correlaciona bien con esfuerzo de desarrollo.

En PSP se usan otras medidas también: elementos de BD, páginas de documentos, formas.

El primer paso para definir LOC como conteo estándar es establecer una estrategia de conteo:

- Contar las líneas.
- Contar palabras reservadas.
- Contar las funciones.

Contabilización del tamaño

Cuando se trabaja con un equipo de desarrollo que produce múltiples versiones, para seguir todos los cambios y adiciones hechas al programa y recolectar los datos necesarios para la estimación se debe usar un sistema de contabilización del tamaño.

Usando datos de tamaño.

Las formas principales de usar los datos del tamaño son en la planificación, la administración de la calidad y el análisis de procesos.

Usando medidas del tamaño para la planificación

Para la planificación si se tiene una medida de tamaño definida y datos históricos de tamaño y tiempo se puede estimar el tamaño de un nuevo producto.

El esfuerzo para adicionar o modificar código será a grandes rasgos el mismo, mientras que el esfuerzo requerido para eliminar una línea o incluir una previamente desarrollada será mucho menor.

Para otro tipo de trabajo se tendrán que hacer diferentes selecciones.

En los trabajos de mantenimiento, por ejemplo el esfuerzo requerido para eliminar una línea puede tomar tanto o más tiempo del requerido para adicionar o modificar.

Determinando la calidad del programa

Dividiendo el número de defectos en una fase por el tamaño del programa da la densidad de defectos en esa fase. Con datos de programas similares, se puede usar estimaciones de la densidad de defecto y tamaño para estimar las pruebas, mantenimiento y costo de servicios. Para el cálculo de la densidad de defecto contar solo los códigos adicionados y modificados durante el desarrollo.

Para la calidad de los programas, la densidad de defecto es generalmente medida por LOC.

Para la calidad de los programas, la densidad de defectos es generalmente medida en defectos por 1000 LOC o KLOC. Para productos terminados la medida más conveniente es 1 000 000 LOC o MLOC. Los trabajos de BD, documentos y otros productos son contados por 100 o 1000 elementos.

Calculando productividad

La productividad es medida generalmente como las horas de labor requeridas para hacer una unidad de trabajo. Si se estima el tamaño de un trabajo nuevo y entonces se usa un factor global de productividad para determinar el total de horas requeridas, se obtendrá generalmente un resultado erróneo. La productividad se calcula dividiendo el tamaño del producto hecho por las horas empleadas en producirlo.

Contadores de Tamaño.

Contar los programas aunque sean pequeños consume tiempo y se puede incurrir en errores. Para programas grandes no es práctico por lo que se requieren esencialmente contadores de tamaño automatizados.

Los contadores pueden destinarse para contar el número de prácticamente cada elemento del producto así como la definición del elemento sea precisa y específica. Por ejemplo: los contadores LOC pueden contar líneas físicas o lógicas. También contar automáticamente elementos de bases de datos puede ser un poco más complejo; si un modelo de objeto es accesible, se puede contar seleccionando conjuntos de elementos de la base de datos.

Contadores físicos de LOC

Es el tipo más simple de contadores de tamaños. En este se cuentan todas las líneas del programa excepto los comentarios y líneas en blanco. Una línea de texto que tiene código y comentarios también es contada.

Contadores Lógicos

Funcionan parecidos a los contadores físicos excepto que el paso de contar las líneas es más complejo. Aunque algunos puedan pensar que contar el principio y final de las declaraciones no es una buena idea, es una cuestión de preferencias personal. Se pudiera demostrar, sin embargo, que el tiempo requerido para desarrollar un programa tiene más alta correlación con LOC cuando los pares comienzo-fin fueron contados que cuando no, esa sería una buena razón para usarlos. Esto sería también la manera apropiada para resolver cualquier pregunta sobre el conteo del tamaño.

Actualmente no existe ninguna evidencia que soporte ningún método de conteo sobre ningún otro. Con buenos datos, sin embargo, un estudio de las distintas alternativas, mostraría rápidamente, si alguno, será el mejor para su trabajo. Podría demostrarse que diferentes acercamientos producen aproximadamente el mismo resultado. En ese caso la selección es arbitraria. El hecho que sea arbitrario, sin embargo, no significa que cada cual deba hacerlo diferente. Si todos los proyectos de una organización usan el mismo conteo estándar, ellos pudieran desarrollar un mismo contador automático con calidad para todos. Tendrían también un largo volumen de datos para usar en las planificaciones y los análisis de calidad.

Contar los elementos del programa

Siempre es deseable medir el tamaño de varias partes de programas largos. Por ejemplo, si deseas rehusar algunas clases o procedimientos nuevamente, te gustaría medir el tamaño de cada uno. De manera similar los contadores ayudan en la estimación.

Para mantener contadores separados para cada clase o procedimiento, se debe determinar cuando comienza y termina cada uno. Debido a que la vía de hacerlo depende del lenguaje de programación, no existe una guía general. Con un poco de estudio se puede llegar a determinar los indicadores de principio y final de los métodos, procedimientos, clases u otros elementos del lenguaje que se use. Se puede usar la definición para escanear los textos del programa e iniciar y parar el contador de tamaño en los puntos apropiados.

Estándar de Codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código.

Las ventajas que brinda el uso de estándares de codificaciones son: se reduce la cantidad de errores, garantiza la obtención de un código comprensible, garantiza buena comunicación entre los integrantes del equipo además de facilitar el mantenimiento, rehúso y revisión.

Otras mediciones de tamaño

La medición por líneas de código (LOC) es solo una opción para la medición de tamaño. Como se pudo observar anteriormente, dependiendo del tipo de trabajo que se hace, elementos de bases de datos, correcciones de errores, o páginas pueden ser más apropiadas. Otra medición útil y mucho más general es conocida como puntos de función. La medición de puntos por función fue desarrollada a finales de los 70 y ha llegado a ser ampliamente utilizada. Actualmente existe un grupo de estándares, el grupo internacional de usuarios de puntos de función que utiliza puntos de función, además de la disponibilidad de una cantidad considerable de literatura.

Un punto a recordar es que no hay una forma de medición que sea la mejor para cada situación. Si prefiere LOC, elementos de bases de datos, o puntos de función, la mejor estrategia es obtener datos en la estimación, el tamaño actual y las mediciones de tiempo. Luego realizar un análisis de estos datos para observar cuales mediciones y métodos se ajustan más en cada situación.

1.10 Métodos de estimación de Software.

En principio, las estimaciones son hechas comparando el trabajo planeado con trabajos desarrollados anteriormente. Descomponiendo los productos en piezas más pequeñas y comparando cada parte con datos sobre las partes similares de productos anteriores, se puede determinar el tamaño del nuevo producto. Esta estrategia funciona bien para estimar casi cualquier tipo de trabajo de desarrollo. Sin embargo, requiere datos sobre los productos que se han desarrollado antes y el trabajo requerido para desarrollarlos. También se necesita un método que utilice datos históricos para hacer las estimaciones.

Diseño Conceptual

Según PSP, se debe primero estimar tamaño y luego hacer la estimación referente al tiempo de desarrollo. Para que la estimación sea certera debe estar basada en un diseño conceptual inicial donde este refleje el modo en que vamos a planear y construir el producto. El diseño conceptual define un diseño preliminar con el nombre de cada una de las partes que componen el producto y sus funciones. Ahora, no se debe producir un diseño conceptual completo durante la planeación, solamente deben definirse las partes y las funciones que integraran el producto.

Para lograr estimaciones confiables se debe refinar el diseño conceptual a un nivel donde se conozcan las partes que se van a construir. Luego, se chequea con los datos históricos de proyectos desarrollados anteriormente si existen partes similares a las del diseño. Si las partes no coinciden se debe entonces refinar el diseño conceptual a un nivel apropiado. [17]

El Método de estimación basado en Proxy (PROBE)

El método PROBE se desarrolló con estos fines, determinar el valor estimado del tamaño total de líneas de código (LOC) de un programa y el tiempo que tomó para ello. Esto, se logra por el método de regresión lineal, su fórmula matemática se describe a continuación:

Tiempo de Desarrollo = β0 + β1 * Tamaño Estimado (E)

Estos valores de regresión β son calculados específicamente por el método PROBE. Este método ha sido automatizado en la herramienta ProcessDashBoard. [17]

Valor Ganado (EV)

Cuando se planea un proyecto de muchas tareas es necesario seguir y reportar el progreso, fundamentalmente cuando las tareas son completadas en orden diferente al que se planeó originalmente. Si sólo se tuviese unas pocas tareas o estas se desarrollaran en iguales intervalos de tiempo, esto no sería difícil, sin embargo los proyectos generalmente tienen muchas tareas de diferentes tipos y tamaños.

A cada tarea se le debe asignar un tiempo (valor planificado o PV) y cuando la tarea termina el valor planificado se convierte en valor ganado. Este valor se agregará al valor ganado de todo el proyecto, si una tarea es muy grande debe dividirse en subtareas para poder medir el progreso.

La medida del valor ganado provee una vía conveniente para dirigir el seguimiento del progreso y reportar los problemas. Siempre que se complete una tarea se obtendrá una cantidad de EV, para juzgar el progreso contra el plan se debe simplemente comparar el EV de cada semana con el valor planeado para esa semana, si el EV acumulado es igual o excede el PV acumulado entonces no habrá retraso.[17]

1.11 Plantillas de Diseño de PSP.

PSP propone 4 plantillas para representar el diseño de un programa. Estas forman parte de los formularios de PSP 2.1. Las mismas son usadas para describir las propiedades esenciales y la estructura de los módulos de los programas son utilizadas para minimizar los duplicados. Cada

elemento es guardado en un solo lugar y su localización es referenciada cuando es necesario. Esto ahora tiempo, reduce la probabilidad de error y proporciona referencias confiables. Un buen diseño debe tener el mínimo de redundancia.

Aunque muchas de las propiedades de los sistemas de software no están cubiertas por estas plantillas, estas propiedades deben ser reflejadas en el diseño o el código de uno o más módulos del programa. Estas plantillas ayudan a asegurar que estas propiedades son implementadas completa y apropiadamente en el nivel del módulo. [18]

Plantilla de especificación operacional (OST)

La OST es una forma simplificada de usar casos que son usados para describir el comportamiento operacional de un programa. Ayuda a visualizar como el programa debe reaccionar bajo varios escenarios de uso. Cuando se enfrenta una decisión de diseño que involucra un actor, esto produce un escenario de ensayo para ver cómo cada acción puede aparecer par el actor. Esto puede ayudar a visualizar el comportamiento del programa para hacer mejores selecciones del diseño.

Plantilla de especificación funcional (FST)

Esta plantilla proporciona una forma simple de documentar muchos de los materiales de los diagramas de clase de UML. El FST describe una parte, incluido los métodos, sus relaciones y sus restricciones. La parte puede ser una clase, un módulo de programa o incluso un programa grande o un sistema. La parte o nombre de la clase se lista en la parte de arriba, junto con la clase u otras partes de las cuales desciende directamente. Los atributos se listan a continuación, con sus declaraciones y descripciones. La tercera sección lista la parte de los métodos o elementos, con sus declaraciones, descripciones y lo que devuelve.

Plantilla de Especificación de Estado (SST)

El comportamiento de estado de una máquina se define por sus actuales entradas y el estado del sistema. La plantilla Especificación de Estado proporciona una forma simple para describir de manera precisa el comportamiento del estado de una parte.

Plantilla de Especificación Lógica (LST)

Esta plantilla muestra cómo usar un lenguaje simple de programación del diseño para describir la lógica interna de una clase. PDL a veces es llamado como seudocódigo. El LST proporciona una descripción en seudocódigo del programa. Su objetivo es explicar concisa y claramente qué debe hacer el programa y cómo.

Uso de las plantillas de PSP para el diseño

EL principal intento de las plantillas de PSP para el diseño es guardar el diseño, no ayudar a producir el diseño. Las plantillas definen de manera precisa qué debe hacer el programa y cómo debe ser implementado. Las elecciones del método de diseño son particularmente importantes porque el diseño es un proceso intelectual y es difícil hacer un buen trabajo intelectual en un lenguaje o con métodos que no son fluidos. Por esto, por lo menos hasta que se hayan usado estas plantillas por un tiempo y se esté familiarizado con ellas se debe continuar produciendo diseños como se hayan hecho anteriormente pero guardándolos en las plantillas. Completando las plantillas se identificarán generalmente problemas de diseño. Cuando se gane experiencia con las plantillas se verá que también estas ayudan a producir el diseño.

Estas plantillas fueron hechas tan genéricas como fue posible de manera que puedan ser usadas con cualquier método de diseño. Si el método de diseño que se está usando también produce la información requerida por una o más plantillas, no es necesario gastar tiempo duplicando ese material. Antes de decidir omitir una o más de estas plantillas, sin embargo, se debe asegurar que el método de diseño usado captura toda la información requerida en los detalles especificados por las plantillas.

1.12 Descripción del dominio del problema.

En la investigación fue necesario profundizar sobre el funcionamiento de los proyectos del Polo Productivo PetroSoft en la facultad, con el objetivo de conocer el estado actual y los problemas existentes en los mismos. El Polo Productivo PetroSoft está dividido en tres proyectos fundamentales, estos son el de Conceptualización de Soluciones, el Ministerio de Energía y Petróleo (MEMPET) y el Sistema Automatizado para el Control de la Gestión de Indicadores de las Refinerías (SACGIR). En este proyecto no se aplica correctamente el sistema de planificación, para la gestión del software y existen problemas en la estimación de riesgos. Se trabaja sobre estimaciones ficticias.

Es preocupante el descontento general del equipo de desarrollo, problema fehaciente que se evidencia también cuando se realizan los cortes de proyecto verificando gran parte de incumplimiento de tareas específicas, causando de esta manera principalmente el atraso general del proyecto. Tomando como referencia el proyecto Sistema Automatizado para el Control de la Gestión de Indicadores de las Refinerías (SACGIR) del Polo Productivo PetroSoft, se demuestra la necesidad de un proceso de control del trabajo individual y en equipo para aplicar entre los desarrolladores del mismo, así como de una selección más acertada de las herramientas y metodologías a utilizar para el desarrollo y puesta en práctica del software. Teniendo en cuenta los resultados de este proyecto se

reflejan de forma resumida, cerca de dos meses de atraso. Existe un descontento generalizado entre los desarrolladores y jefes del proyecto a causa de la labor intensa poco orientada a necesidades y normas reales tanto del proyecto como de los mismos miembros del equipo de desarrollo, y en general una labor muy pobre de los sistemas de monitoreo, control, auditores, probadores y referente al personal en general vinculado con la calidad de este proceso productivo.

El proceso de gestión de los recursos, describe mediante organigramas la distribución tanto de las máquinas, como los roles de los miembros del proyecto, almacenando los datos relacionados con el personal, en un documento llamado SACGIR Capital Humano y recursos materiales almacena los datos relacionados con las máquinas asignadas a los polos, el nombre del polo al que se hace referencia, la cantidad de máquinas según el tipo de memoria, se almacena los datos relacionados con los recursos asignados al proyecto, la cantidad de máquinas portátiles y la cantidad de dispositivos externos, registrado en el documento SACGIR Recursos Materiales. El líder del proyecto cuenta con los conocimientos necesarios para llevar a cabo la realización del proyecto. La visión del proyecto es discutida con todos lo miembros del equipo y es registrada a través de informes de reuniones del consejo de guerra en el cual es analizada la situación del proyecto, dándole cumplimiento al chequeo de acuerdos, asignando responsables en cada uno de los puntos a debatir, aunque existe un atraso, ya que el último informe registrado de reunión del Consejo de Guerra fue el día 7 de enero de 2008.

Los miembros del equipo reciben una adecuada superación a través de cursos de superación y cursos optativos tales como Ingeniería de Requisitos, Modelado de Negocio y Cake PHP. Existen problemas en la planificación, esta no se realiza del todo y existe mucho atraso en cuanto a la actualización de documentos. Se planifica por fases y por iteraciones del ciclo de vida del proyecto, aunque no todas las fases o procesos del ciclo de vida están reflejadas en la planificación, como es el case del proceso de pruebas del software que no están registradas en los cronogramas, no se establecen puntos de chequeo que permitan verificar el estado del proceso.

Los objetivos son medibles y claros por fases e iteraciones. No existe un documento en el que se refleje el cumplimiento de las actividades actualizado, el último informe de actividades registrado es del día 24 de enero de 2008. No se realiza una planificación basada en los datos obtenidos del control de tiempo y esfuerzo de los miembros del equipo.

No se registra el tiempo de trabajo personal de los miembros del equipo, se hace una distribución del tiempo por miembros en cuanto a uso de las PC's, asignación de tiempo de máquina y se verifica su cumplimiento pero no se registra el tiempo de trabajo personal en ningún documento, por tanto no se registra de forma personal los defectos encontrados, no se lleva un control personal de los

compromisos documentados , pero se realizan controles con el objetivo del cumplimiento de las tareas asignadas , se registran a través de documentos el control de la asistencia aunque existe atraso en cuanto a esto , el último informe sobre el control de asistencia registrado es del día 22 de noviembre de 2007 y se lleva a cabo un proceso de evaluación de los miembros del proyecto, y se procederá a realizar un análisis disciplinario con los estudiantes evaluados de Mal y de Bajo rendimiento, con sus respectivas sanciones. No existe por parte de los miembros del equipo conocimientos sobre los modelos de calidad, pero si está registrado y bien definido el plan de aseguramiento de la calidad, nombrado SACGIR Plan de Aseguramiento de la Calidad v_1.0 fecha 28 de febrero de 2008.

1.13 Conclusiones.

Es importante destacar que la calidad del software está estrechamente relacionada con la buena aplicación del Proceso Personal de Software (PSP), ya que logrando una estrategia correcta se logra un producto con la calidad requerida y en el tiempo establecido. Muchos son los factores que influyen en el resultado final del producto, de ahí la importancia de un correcto, organizado y bien estructurado trabajo personal, constituyendo la primera base en la línea de la arquitectura del software.

Se puede señalar que los proyectos del Polo Productivo PetroSoft de la facultad 9 se encuentran en un estado básico en cuanto a calidad y uso del PSP se refiere, trayendo como consecuencias problemas en la aplicación de sistemas de planificación para la gestión del software, existiendo un descontento general por parte del equipo de desarrollo, causando el atraso del proyecto. La descripción antes mencionada, encamina a una fuerte investigación para darle solución a estos problemas.

CAPÍTULO 2

2.1 Introducción.

El presente capítulo tiene como objetivo proponer una solución a la situación anteriormente explicada en la descripción del dominio del problema. Después de un análisis sobre la situación de los proyectos que se realizan en dicho polo, tomando como muestra el Sistema Automatizado para el Control de la Gestión de Indicadores de las Refinerías (SACGIR), se pretende desarrollar una estrategia para aplicar correctamente el Proceso Personal de Software (PSP). La estrategia consiste en hacer una revisión de todos los documentos que genera el PSP, desglosándolos por cada uno de sus niveles y con sus fases correspondientes.

Este análisis se hará explicándose en que consiste el documento (enmarcado con su nivel y fase correspondiente), así como su importancia y por último se recomienda como mejorarlo, o sea plantear la estrategia de solución, mediante el uso de plantillas, sólo las propuestas, con el objetivo de optimizar el uso de PSP y hacerlo menos engorroso.

2.2 Guiones de PSP0.

Los guiones de PSP son aquellos que guían a través de los pasos de procesos. El nivel PSP0, está compuesto por tres partes fundamentales, el criterio de entrada, en el cual se localiza la descripción del problema, formularios LOGT, LOGD y el Resumen del Plan del Proyecto, consta de tres fases fundamentales, estas son, planificación, desarrollo y postmortem, y por último el criterio de salida en el que se puede obtener el programa ya probado exhaustivamente, los LOGT y los LOGD completos y el Formulario del Resumen Plan del Proyecto lleno con datos estimados y reales.

2.2.1 Fase de Planificación de PSP0.

Cada una de las diferentes fases de PSP0, planificación, desarrollo y postmortem, están asociadas a un grupo de tareas, por ejemplo en la fase de planificación, que es donde se planifica el trabajo y se documenta el plan, tiene asignadas un grupo de tareas, en primer lugar obtener los requerimientos,

estimar el tiempo de desarrollo, entrar los datos en el plan usando el formulario Resumen del Plan del Proyecto.

Para la obtención de buenos requerimientos se debe saber que estos deben ser especificados por escrito, como todo contrato o acuerdo entre dos partes, posibles de probar o verificar , descritos como una característica del sistema a entregar , o sea centrarse en qué es lo que el sistema debe hacer y debe ser lo más abstracto y conciso posible, asegurarse de que son claros y no ambiguos, una vez culminada de forma correcta y realizado un levantamiento de requerimientos se pasa a la siguiente tarea la de estimar recursos.

La estimación de recursos debe asegurar que se estimó en el mejor tiempo para la terminación del programa, actualizando de forma correcta y continua, en el registro de tiempo, todas las actividades realizadas, obteniendo como resultados de esta fase requerimientos bien documentados y el formulario Resumen del Plan del Proyecto con el tiempo de desarrollo estimado, sólo haciendo esto es que se puede pasar a la siguiente fase para dar como culminada la fase de planificación y centrarte en la fase de desarrollo .

2.2.2 Fase de Desarrollo de PSP0.

En la fase de desarrollo cuando se culmina el levantamiento de requisitos, se procede al diseño del programa, posteriormente se implementa este diseño, y se compila, se ajusta y se registran los defectos en LOGD y se completan los LOGT. La misma consta de cuatro subfases tales como son, diseño, codificación, compilación y prueba. En la subfase de diseño, se diseña el programa, se revisan los requerimientos y se elabora un diseño acorde a ellos, se debe realizar un buen análisis del programa identificando diagramas de clases del análisis por casos de uso, diagramas de clases de diseño, diagramas de interacción (secuencia y colaboración) y por último se registra el tiempo en LOGT para dar culminada la subfase de diseño y pasar a la implementación de este, o sea la subfase de codificación.

En codificación es donde se implementa este diseño, se define la organización del sistema en subsistemas, se implementa los elementos de diseño, se integran los resultados en un sistema ejecutable, se registra en LOGD, cualquier defecto de requerimientos y de diseño, para culminar esta subfase se registra el tiempo en LOGT.

En la siguiente se procede a la compilación, que es donde se compila el programa, se repara y registran los defectos encontrados, se compila hasta corregir los errores y se van eliminando los defectos encontrados, se registran los defectos en LOGD y el tiempo en LOGT, culminando así esta

subfase, hasta que no se realice correctamente todo lo anteriormente descrito no se avanza hacia la próxima fase.

En la fase de prueba, se prueba el programa, se repara y registran los defectos encontrados. Se identifican los métodos de prueba y se realizan los casos de pruebas. Cuando culmina la prueba, el resultado de la fase de desarrollo será un programa probado exhaustivamente, LOGT y LOGD completo, se procede a la fase de postmortem.

2.2.3 Fase de Postmortem de PSP0.

En la fase de postmortem que es donde se registra el tiempo actual, defectos y tamaño de datos en el plan, tiene como objetivo fundamental completar el Resumen del Plan del Proyecto con datos reales de tiempo, tamaño y defecto. Consta de tres tareas fundamentales como son, determinar los defectos inyectados, los defectos eliminados y revisar los registros de tiempo.

Estas tareas parten, de un Resumen del Plan de Proyecto con tiempo planificado, los LOGT y LOGD completos y de un programa corrido y probado. La tarea Defectos Inyectados, determina de los LOGD el número de defectos inyectados en cada fase y se registra en Defectos Inyectados en la columna real del Resumen del Plan del Proyecto. La de Defectos Eliminados lo que hace es que determina el número de defectos eliminados en cada fase y se registra en defectos eliminados en la columna real del Resumen del Plan del Proyecto y por último en la revisión del tiempo se verifica si el LOGT está completo y se entra el tiempo total gastado en cada fase del proyecto en el Resumen del Plan del Proyecto, obteniéndose como resultado de la fase de postmortem un programa probado completamente, un Resumen del Plan del Proyecto , LOGT y LOGT completos.

2.2.4 El Registro de Tiempo de PSP0.

El Cuaderno de Registro del tiempo de PSP0 esta compuesto por los siguientes datos básicos: el proyecto o los programas en los que se esta trabajando, las fases del proceso para las tareas, la fecha del tiempo en el que se inició y culminó cada tarea, las interrupciones de tiempo, la variación del tiempo trabajado en la tarea, así como sus comentarios.

Tiene como objetivo registrar el tiempo gastado en cada fase del proyecto, se registra también todos los tiempos gastados en el proyecto en minutos, siendo lo más exacto posible y si se requiere espacio adicional se recomienda usar otra copia del formulario, usando estos datos para completar el Resumen del Plan del Proyecto.

Entre los principales campos del cuaderno se encuentran los siguientes, encabezamiento campo en el cual se guardan el nombre, la fecha, el profesor y el número del programa, en el campo fecha; se guarda la fecha en que se registra el tiempo; en los campos inicio y fin se encuentran los tiempos en que se comienza y se deja de trabajar en la tarea; en tiempo de interrupción se encuentra cualquier interrupción que ocurre durante la tarea y la razón de esta; en delta tiempo, tiempo real gastado en la tarea fecha de inicio ,fin y la interrupción ; en el campo fase, el nombre de la fase en la que se esta trabajando y por último cualquier comentario interesante. Una propuesta a la plantilla anteriormente descrita es la siguiente:

Nombre: _____ **Fecha:** _____

Profesor: _____ **# Programa:** _____

Fecha	Inicio	Fin	Tiempo interrupción	Delta tiempo	Fase	Comentarios

Tabla 2. Plantilla propuesta del Cuaderno de Registro de Tiempo de PSP0

La importancia de este cuaderno es que permite acceder a gran cantidad de información histórica sobre la utilización del tiempo. Conocer los tiempos permite realizar estimaciones seguras que repercuten a la hora de la entrega del producto final, un ahorro en tiempo y dinero, implica un aumento de la calidad del trabajo que se realiza.

2.2.5 El Registro de Defectos de PSP0.

El Cuaderno de Registro de defectos de PSP0 permite registrar cada defecto encontrado y corregido en el proceso de desarrollo. Sirve como información para elaborar el proceso de depuración, almacena los datos de los defectos encontrados, cuando se encuentra un error y se decida repararlo se debe comenzar a contar el tiempo del arreglo una vez que ha sido reparado se debe entrar todos los datos para dicho defecto.

Tiene como objetivo registrar cada defecto cuando lo encuentra y lo corrija, se registran todos los defectos encontrados en revisión, compilación y prueba, usando estos datos para completar el Resumen del Plan del Proyecto.

Entre los principales campos se encuentra el encabezamiento, campo en el cual se guardan el nombre, la fecha, el profesor y el número del programa; en el campo fecha; se guarda la fecha en que

se registra el tiempo; en el campo número, por cada programa debe ser un número consecutivo comenzado por 1 o 001; en Fase inyectada (F Inyectado) en la cual el defecto fue inyectado ;en el campo tipo se selecciona el mejor de los 10 tipos estándares , para registrar el tipo de defecto se debe entrar el número que aparece en el Estándar de Tipo de Defecto el cual es desarrollado de acuerdo las necesidades de cada ingeniero; en la fase eliminado (F Eliminado) en la cual el defecto fue eliminado (generalmente donde es encontrado); en tiempo de corrección (T Corrección) se guarda el tiempo en corregir el defecto ; en defecto corregido (D Corregido) , si se inyectó este defecto mientras se corregía otro , registrar el número de defecto corregido inadecuadamente , se identifica el número del defecto y por último en el campo descripción, se guarda el número y una breve descripción del defecto.

Se registran los datos de cada defecto cuando es reparado, pues tiene gran importancia para cuando se trabaje en equipo TSP (Proceso de Software en Equipos), estos datos de defectos sirven de ayuda para producir consistentemente productos que tengan escasos defectos después de entregado al cliente, ganando así en tiempo para realizar las pruebas. Para ahorrar dicho tiempo, se tendrá que formar buenos hábitos de registrar defectos durante PSP y debe continuar registrando y analizando defectos cuando se trabaje en equipos TSP. Una propuesta a la plantilla anteriormente descrita es la siguiente:

Nombre: _____ **Fecha:** _____

Profesor: _____ **# Programa:** _____

Fecha	#	F Inyectado	Tipo	F Eliminado	T Corrección	D Corregido	Descripción

Tabla 3. Plantilla propuesta del Cuaderno de Registro de Defectos de PSP0

La utilización del Cuaderno de Registros de Defectos tiene gran importancia ya que contribuye a mejorar la programación, ya que entender los defectos es algo importante para aprender a gestionarlos y mejorarlos. A reducir su aparición, al aprender a gestionar defectos implica reducir su aparición. Al ahorro del tiempo, los errores tienden a provocar más errores (un error de diseño causará uno o más errores en implementación), eliminarlos a tiempo implica no tener que corregirlos de nuevo. Ahorra dinero, encontrar y corregir un defecto es por lo general caro, minimizar y corregir un defecto supone un gran ahorro económico. Hacer el trabajo responsablemente puesto que es responsabilidad del ingeniero de software encontrar y corregir los defectos.

2.3 Guiones de PSP0.1

Los guiones de PSP0.1 son muy parecidos a los de PSP0 sólo se le agregan algunos aspectos, quedando demostrado la estructura incremental del PSP. Aparece en la fase planificación un nuevo objetivo, estimar LOC nuevas y cambiadas según van siendo requeridas. En la fase de postmortem aparece el conteo de LOC del programa entero, así como la determinación de LOC base, eliminado y adicionado. Es también introducido un formulario para registrar propósitos de mejora de procesos (PIP), formulario que permite a los ingenieros contar con una vía conveniente para registrar los problemas del proceso y las soluciones propuestas.

2.3.1 Fase de Planificación de PSP0.1

En la fase de planificación de PSP0.1 se mantienen las mismas tareas, la obtención de requerimientos, la estimación de recursos, y la estimación de tiempo. En la estimación de tiempo se le agrega un nuevo propósito, el de asegurar la mejor estimación de LOC nuevas y modificadas para el programa y en la estimación de recursos, el uso del por ciento hasta la fecha para la mejora de la planificación de la fase siguiente.

2.3.2 Fase de Desarrollo de PSP0.1

En la fase de desarrollo de PSP0.1 se sigue con la misma idea de que a partir de que los requerimientos estén bien definidos, con los Estándares Tipo Defecto y Codificación y con el Resumen del Plan del Proyecto con el tiempo de desarrollo y tamaño del programa estimado se obtengan programas probados exhaustivamente, conforme al estándar de codificación y los formularios LOGT y LOGD completos.

2.3.3 Fase de Postmortem de PSP0.1

En la fase de postmortem PSP0.1 se mantiene con las mismas tareas, la determinación de defectos inyectados y eliminados. Se sigue revisando si el LOGT está completo y si se guarda el tiempo total gastado en cada fase. En cuanto a contabilización del tamaño se trata, aparece el conteo de las LOC del programa entero, se determina las LOC base, eliminado, reusado, modificado, adicionado, total y total de nuevos modificados y de nuevos reusados.

2.3.4 Propuesta de Mejora del Proceso (PIP) PSP0.1

El formulario de Mejora del Proceso (PIP), provee un registro de problemas encontrados y mejoras para el proceso. Describe los problemas encontrados en cada proyecto. Contiene los datos estimados y reales del proyecto en un formato conveniente y legible, el objetivo fundamental es registrar las ideas sobre las mejoras en el proceso, estableciendo prioridades en la planificación y registrando las lecciones aprendidas.

Este formulario se usa para establecer prioridades en el plan de mejoras, registrar ideas sobre mejoras en el proceso y registrar lecciones aprendidas y soluciones iniciales. Consta de un encabezamiento, en el cual se encuentra el nombre, la fecha, el profesor, el número del programa y el nombre del proceso utilizado.

Existen dos casillas fundamentales la del problema y la de la propuesta, en la del problema se encuentra una descripción detallada del mismo, tan claro como sea posible, se deja registrado el grado de dificultad encontrado y el impacto en el producto, se enumera los problemas en cada formulario y se usa un número de secuencia conveniente comenzando con 1 en cada formulario.

En la casilla propuesta, se describe la mejora del proceso propuesta tan explícita como sea posible, se referencia el elemento específico del proceso y el número del problema, si es importante se debe describir la prioridad.

Por último se registra en la casilla comentarios, las lecciones aprendidas (Lec Aprendidas), cualquier condición que requiera más tarde ser recordada debido al buen o mal trabajo del proceso y para cada proyecto completar al menos un formulario PIP con comentarios sobre el proceso. Una propuesta al formulario anteriormente descrito es la siguiente:

Nombre: _____ **Nombre del Proceso:** _____

Profesor: _____ **# Programa:** _____ **Fecha:** _____

Problema				**Propuesta**		**Cometarios**
Descripción	**Grado Dificultad**	**#**	**# Secuencia**	**Descripción**	**Prioridad**	**Lec Aprendidas**

Tabla 4. Propuesta del Formulario Mejora del Proceso (PIP) de PSP0.1

2.4 Guiones de PSP1.

Este guión parte de la descripción del problema, de un formato de estimación del tamaño, de datos estimados históricos y tamaño real, de Log de tiempo y de defecto, de un estándar de tipos de defectos y de un cronómetro (opcional).

En la fase de planeación se obtienen los requerimientos y se utiliza el método PROBE, para estimar el total de LOC nuevas, se completa el LOGT, se estima el tiempo requerido para el desarrollo y se completa el formato de estimación del tamaño, culminando así la fase de planeación, una vez cumplido con esto se puede pasar a la siguiente fase, que es la de desarrollo.

En la fase de desarrollo se diseña el programa, se implementa el diseño, se compila, registran, se arreglan los defectos encontrados y por último se prueba el programa, hasta que este no se pruebe no se puede pasar a la fase de postmortem.

En la fase de postmortem se completa el Plan del Proyecto con los datos actuales de tiempo, defecto y tamaño, obteniendo como resultados un programa al que se le realizaron pruebas, LOG de tiempos y defectos completos, formas de PIP completas, formato de Estimación de Tamaño y formato de Reporte de Pruebas completos, y un Plan de Proyecto con los datos estimados y actuales.

PSP1 dispone de un guión encargado de dirigir todo el proceso referente a este nivel, consta de tres fases, planeación, desarrollo y postmortem, aparecen dos plantillas, la Plantilla de Estimación de Tamaño, el Método de Estimación PROBE y la Plantilla de Reporte de Prueba.

2.4.1 Método de Estimación PROBE.

Según PSP se debe primero estimar tamaño y luego hacer la estimación del tiempo. Para que la estimación sea certera debe estar basada en un diseño conceptual inicial donde se refleje el modo en que se va a plantear y construir el producto. Para lograr estimaciones confiables se debe refinar el diseño conceptual a un nivel que se conozcan las partes que se van a construir, se chequea con los datos históricos desarrollados anteriormente y si las partes no coinciden se refina el diseño conceptual a un nivel apropiado.

Posteriormente a través del proxy, se relaciona el tamaño del producto con las funciones del mismo. Los proxies ayudan a juzgar efectivamente el tamaño de un producto tomándolos como unidades de medida. Algunos ejemplos de proxies son las clases, las tablas, las filas, los puntos de función, los objetos, componentes, informes, guiones y ficheros.

Se propone utilizar el método de estimación basado en proxy (PROBE), pues este es el encargado de determinar el valor estimado del tamaño total de líneas de código (LOC) y el tiempo que se tomó para ello. Esto se logra por el método de regresión lineal a través de la fórmula matemática:

Tiempo de Desarrollo= β0 + β1 * Tamaño Estimado (E).

Para la aplicación del método PROBE se deben seguir los siguientes pasos:

Primero, comenzar con el diseño conceptual, identificar objetos, calcular LOC modificadas y proyectadas, estimar tamaño del programa, calcular intervalo de predicción y por último determinar el tamaño y tiempo de intervalos de predicción.

2.4.2 Plantilla de Estimación de Tamaño de PSP1.

El diseño conceptual constituye la base fundamental de la Plantilla de Estimación de Tamaño, la cual se relaciona con el método PROBE. Se propone realizar la plantilla de estimación de tamaño, en caso de que el desarrollo sea una mejora o modificación de un programa existente que se debe hacer, se debe definir LOC del Programa Base, Tamaño Base (B), LOC Eliminadas (D) y LOC Modificadas(M) .

En primer lugar se debe contar y registrar el tamaño del programa base en B, registrar las LOC del programa base que se van a eliminar en D, registrar las LOC del programa base que se van a modificar en M.

En caso que en este desarrollo se piense adicionar LOC al programa base, se deben definir LOC Objetos adicionales a la base (BA), se deben identificar las funciones que se van a adicionar a objetos existentes, estimar las LOC por cada función adicional, en caso de ser apropiado estimar esta función como si se tratara de un nuevo objeto, es decir utilizando el tipo de categoría al cual aplica y se deben registrar el total de LOC en BA.

En caso de nuevos objetos se definen LOC Objetos nuevos (PA), se asigna un nombre al nuevo objeto planeado a la base, se estima el tipo de objeto y el número de métodos que el objeto contendrá, se estima el tamaño relativo del objeto, se determina el valor estimado de LOC y se totaliza el estimado de LOC de Objetos nuevos.

En caso de objetos reutilizados, definir Total reusados (R), registrar el nombre de cada objeto reusado sin modificar, registrar las LOC de cada objeto reusado sin modificar, sumar estas LOC y registrar este valor en Total Reusadas(R). Para el calculo del tamaño se debe definir LOC estimada de objetos (E), sumar las LOC de las adiciones a la base (BA), las LOC de los objetos nuevos (PA), y las LOC modificadas (M) y registrar este total en LOC estimadas de objetos (E). Usando PROBE se calculan

los parámetros de regresión de tamaño y tiempo, y posteriormente se realizan los cálculos del intervalo de predicción.

2.4.3 Formulario de Reporte de Prueba de PSP1.

PSP1 introduce el Formulario o Plantilla de Reporte de Prueba, cuyo objetivo es mantener un registro real de las pruebas que se le hacen a cada programa y de los resultados de estas pruebas, para lograr tener un registro que sea fiable en el futuro. Se pretende que al final estas pruebas sean lo suficientemente completas para repetirlas cada vez que sea necesario y obtengan los resultados que se deseen. El uso correcto de este formulario ayuda al programador a simplificar la aplicación de pruebas a los programas que sufren de alguna modificación.

Este formulario tiene como propósito mantener un registro de las pruebas ejecutadas y los resultados obtenidos, el reporte debe ser lo suficientemente completo de tal forma que se pueda utilizar más adelante, para repetir las mismas pruebas y obtener los mismos resultados. La utilización correcta de este reporte simplifica la prueba de regresión, de los programas modificados.

Consta de un encabezado, en el que se encuentra el nombre del programador y la fecha actual, el nombre del instructor y la fecha del programa. El nombre y número de la prueba, en el que se identifican de manera única las pruebas de ejecución de cada programa que pueden ser las mismas pruebas con datos diferentes, los mismos datos con diferentes pruebas y las mismas pruebas que se vuelven a ejecutar después del arreglo.

Posteriormente se listan los resultados esperados y actuales. En los esperados se listan los resultados que la prueba debería generar si se ejecuta en forma apropiada y en los actuales se listan los datos que actualmente se produjeron, cuando una prueba se ejecuta múltiples veces para arreglar múltiples defectos se debe anotar los resultados de cada prueba. Una propuesta al formulario anteriormente descrito es la siguiente:

Nombre: _____ **Fecha:** _____

Profesor: _____ **# Programa:** _____

Nombre prueba	#	Objetivos	Condiciones	Resultados esperados	Resultados actuales

Tabla 5. Propuesta del Formulario de Reporte de Prueba de PSP1

2.5 Guiones de PSP1.1

En PSP1.1 se sigue con la misma idea de los guiones anteriores de PSP, pero se le agregan algunas actividades importantes, en la fase de planeación específicamente, en la cual se utiliza el método PROBE para estimar tamaño, aparecen el Plan de Tareas y el Plan de Cronograma. En la tarea estimación de recursos, se utiliza el método PROBE para estimar el tiempo requerido para desarrollar el programa, surge una nueva tarea, la planeación de tareas y cronogramas, en la cual se deben completar los formatos de planeación de tareas y cronograma para los proyectos que requieran de varios días o mucho trabajo, culminando esta fase con los formatos de planeación de tareas y cronogramas completos, sólo así se procede a la siguiente fase de desarrollo.

En la fase de desarrollo teniendo en cuenta estos formatos, la planeación de tareas y cronogramas completos, se obtienen como resultados un programa completamente probado y que satisface el estándar de codificación, y un Reporte de Prueba completo, condición suficiente y necesaria para pasar a la siguiente y última fase, la de postmortem.

En la fase de postmortem, se obtienen como resultados, un programa completamente probado, Log de tiempo y defectos, completos, un Reporte de Pruebas completo, un Plan de Proyecto completo, y formas de PIP, completas que describen: Problemas del proceso, sugerencias para el mejoramiento y lecciones aprendidas.

2.5.1 Valor Ganado (EV).

Para poder hacer una buena planeación de tareas, es necesario tener conocimiento acerca del significado del Valor Ganado (EV). Cuando se planea un proyecto que tiene muchas tareas, y estas se van completando en orden diferentes en el que se planeó, se debe reportar el progreso. A cada tarea se le asigna un tiempo (valor planificado PV), al terminar esta se convierte en valor ganado, este valor se le agrega el valor ganado de todo el proyecto. Si existen tareas muy grandes se pueden dividir en subtareas para poder medir el progreso. La medida del valor ganado provee una vía conveniente para dirigir el seguimiento del progreso y reportar problemas. Siempre que se complete una tarea se obtendrá una cantidad de EV para juzgar el progreso contra el plan, se debe simplemente comparar el EV de cada semana con el valor planeado para esa semana, si el valor acumulado es igual o excede al PV entonces no habrá retraso.

2.5.2 Plan de Tareas de PSP1.1

El propósito de este plan es estimar el tiempo de desarrollo para cada tarea del proyecto, calcular el valor planeado para cada tarea del proyecto, estimar la fecha de finalización para cada tarea y proporcionar una base para hacer seguimiento del progreso del cronograma aún cuando las tareas no terminan en el orden planeado.

En este plan se nombran y enumeran consecutivamente las tareas a desarrollar como parte del proyecto, se recomienda que las tareas deben listarse en el orden en que se esperan sean terminadas, el nombre de las tareas tiene que expresar un criterio de terminación claro. Se debe registrar las horas planeadas para cada tarea, para cada tarea calcular el Valor Planeado (VP) como: Horas estimadas para la tarea / Total de horas planificadas para todas las tareas. Para cada tarea calcular las Horas Acumuladas (H Acumuladas) como: Horas estimadas para la tarea + Horas acumuladas para la tarea anterior. Para cada tarea calcular el Valor Planeado Acumulado (VP Acumulado) como: Valor planeado para la tarea + Valor planeado acumulado hasta la tarea anterior. Por último se calcula el Valor Ganado Acumulado (EV Acumulado) como: Valor ganado + Valor ganado acumulado de la tarea anterior terminada. La tarea anterior terminada no tiene porque ser la que precede en la lista de tareas. Una propuesta al plan anteriormente descrito es la siguiente:

Nombre: _____ **Fecha:** _____

Profesor: _____ **# Programa:** _____

Tarea		**Plan**					**Real**		
#	**Nombre**	**Horas**	**VP**	**H Acumuladas**	**VP Acumulado**	**Fecha**	**Fecha**	**EV**	**EV Acumulado**
Totales									

Tabla 6. Propuesta del Plan de Tareas de PSP1.1

2.5.3 Plan de Cronograma de PSP1.1

El Plan de Cronograma tiene como objetivos, registrar las horas estimadas y planeadas en términos de período de calendario, relacionar el valor de la tarea planeada con el horario en el calendario y calcular los valores ajustados del valor ganado cuando la tarea finaliza.

Primero que nada desde el inicio del proyecto se recomienda registrar el número de la semana generalmente iniciado en 1, para proyectos más pequeños debe ser más conveniente utilizar días en vez de semanas. Se registra el número de horas directas (HD) que se ha planeado gastar por semana, se recomienda considerar el tiempo que no se trabaja, por vacaciones y considerar otras actividades como reuniones, clases y otros proyectos. Se registra en cada semana el valor de horas acumuladas planeadas (HA).

Cada vez que se termine una semana registrar las horas directas que realmente se dedicaron al proyecto. Para cada semana terminada calcular las horas acumuladas (real) como: Horas directas + Horas acumuladas (real) hasta la semana anterior. Por último, se relaciona con el Plan de Tareas, para determinar el Valor Acumulado Real, se busca en el Plan de Tareas el Valor Acumulado Ganado, en la fecha real de terminación de una tarea cuya fecha sea igual o menor más próxima a la fecha de la semana. Posteriormente calcular el Valor de Ajuste como el Valor Acumulado real de la semana por el Factor de Ajuste de la semana.

Se recomienda si se adiciona una tarea calcular el Factor de Ajuste como: la suma del Valor Ganado de todas las tareas anteriores a la nueva tarea adicionada / (la suma de del Valor Ganado de todas las tareas anteriores a la nueva tarea adicionada + Valor Ganado de la tarea adicionada). Si se elimina una tarea calcular el Factor de Ajuste como: la suma del Valor Ganado de todas las tareas anteriores incluyendo la tarea a eliminar / (la suma del Valor Ganado de todas las tareas anteriores incluyendo la tarea a eliminar - Valor Ganado de la tarea eliminada). Una propuesta al plan anteriormente descrito es la siguiente:

Nombre: _____ **Fecha:** _____

Profesor: _____ **# Programa:** _____

		Plan			Real			
# Semana	**Fecha**	**Fecha**	**HD**	**HA**	**HA**	**HD**	**Valor Acumulado**	**Valor Ajuste**
Totales								

Tabla 7. Propuesta del Plan de Cronograma de PSP1.1

2.6 Guiones de PSP2

En PSP2 se adicionan nuevos elementos en la fase de desarrollo, como son: revisar el diseño y el código, arreglar y registrar todos los defectos encontrados, teniendo como resultados las listas de chequeo de diseño y codificación.

En la fase de planificación de PSP2 en la tarea de estimación de defectos, se recomienda utilizar el valor de los defectos hasta la fecha, por LOC nuevas y cambiadas, para estimar el total de defectos a ser encontrados en el programa, utilizar los datos del por ciento hasta la fecha, para estimar los números de defectos a ser introducidos y eliminados por fase. En la fase de postmortem se determina el rendimiento y se verifica si el valor es razonable y correcto, obteniendo finalmente las listas de chequeo completas para las revisiones de diseño y código.

2.6.1 Control de la calidad personal PSP2

La estrategia de control para la calidad personal consiste en: usar planes y datos históricos para guiar el trabajo, se realiza el diseño y cuando se revise se debe pasar un tiempo suficiente encontrando los defectos más probables, si el trabajo de diseño toma 4 horas, se deben planificar al menos 2 y preferiblemente 2 para hacer la revisión. Para hacer esto productivo se deben planificar los pasos de la revisión basados en la Revisión del Diseño de PSP. Posteriormente se realiza la Revisión de Código propuesto por PSP, haciendo énfasis en tomar el tiempo que dicen los datos históricos, entonces es seguro que se hará un trabajo con calidad.

2.6.2 Esfuerzo personal para la detección de defectos PSP2

El esfuerzo personal para la detección de defectos se puede establecer revisando los datos de los defectos que tanto la persona, como su equipo encontraron durante el desarrollo y las pruebas. Una estrategia posible es centrarse en los tipos de defectos ya sean los encontrados en las pruebas finales del programa, los que ocurren frecuentemente, los que son más difíciles de encontrar o corregir, aquellos para los cuales más rápidamente se pueden identificar acciones de prevención.

El punto clave es revisar cada prueba, reporte de defectos, obtener los datos esenciales y enfocar una estrategia explícita en la prevención, viendo que se trabaja mejor se tiene una mejor idea de cómo proceder. El objetivo de las revisiones es encontrar y corregir defectos, para esto se necesita saber cual es la causa que originó ese defecto para después prevenirlo.

Se recomienda primero seleccionar un tipo de defecto, resumir las causas, buscar patrones o tendencias en los datos que sugieran problemas más complejos o profundos, después se determinan los errores que causan esos defectos, se crean formas de prevenir estos en el futuro, analizando acciones en el pasado y por último se prueban las ideas de prevención de defectos en un programa usando PSP incorporando aquello que funciona en una actualización del proceso.

2.6.3 Revisiones de diseño y código PSP2

Las revisiones de diseño y de código son otros de los aspectos importantes a tener en cuenta cuando se habla de calidad. El proceso PSP toma ventaja del hecho de que los errores de las personas son predecibles. Esto no quiere decir que sean incompetentes, sino que son humanos y como humanos se tiende a repetir los errores.

Se recomienda para hacer revisiones personales, revisar personalmente todo el trabajo propio antes de moverse a la próxima fase de desarrollo. Esforzarse para corregir todos los defectos antes de darle el producto a alguien más. Usar una lista de chequeo personal y seguir un proceso de revisión estructurado. Medir el tiempo de revisión, el tamaño de los productos revisados y el número y tipos de defectos que encuentran. Usar estos datos para mejorar el proceso personal de revisión. Diseñar e implementar sus productos de manera que sean fáciles de revisar. Revisar sus datos para identificar formas de prevenir los defectos, siguiendo estas ideas se mejorará en la productividad personal y se producirá productos sustancialmente mejores.

2.6.4 Lista de Chequeo de Código PSP2

Es recomendable utilizar la lista de chequeo de código pues esta guía una revisión efectiva del código, verifica que el código cubra todo el diseño y que los includes estén completos. La Lista de Chequeo de Código verifica el formato de la llamada a funciones y chequea todas las cadenas que estén identificadas por punteros y terminadas en NULL. Además chequea que cada formato de salida tenga un salto de líneas apropiado. Aquí se verifica el uso apropiado de operadores lógicos y se chequea línea por línea de código según la sintaxis de la instrucción y la puntuación adecuada. Se asegura que el código este acorde al estándar de codificación donde se verifica que todos los ficheros estén apropiadamente declarados, abiertos y cerrados. Una propuesta a la lista anteriormente descrita es la siguiente:

Nombre: _____ **Fecha:** _____

Nombre del Programa: _____**# Programa:** ____

Profesor: _____**Lenguaje:** _____

Objetivo	•Guía una revisión efectiva del código
General	•Una vez que se complete cada paso de la revisión, marcar cada elemento en el cuadro de la derecha •Completar la lista de chequeo para una unit del programa antes de comenzar a revisar la próxima.
Completo	Verificar que el código cubra todo el diseño:
Includes	Verificar que los includes estén completos
Inicializaciones	Chequear la inicialización de variables y parámetros: • Al iniciar el programa • Al comenzar cada lazo • A la entrada de cada función /procedimiento
Llamadas	Chequear el formato de la llamada a función: • Punteros • Parámetros • Uso de &
Nombres	Chequear el nombre letra a letra y su uso: • Su consistencia • Si está en el alcance declarado • Hacer que todas las estructuras y las clases usen la referencia “.”
Cadenas	Chequear que todas las cadenas estén: • Identificadas por punteros • Terminadas en NULL

Punteros	Chequear que: • Los punteros estén inicializados en NULL • Los punteros sean eliminados solo después de new • Los nuevos punteros sean siempre eliminados después de su uso .
Formato de Salida	Chequear en cada formato de salida. • Salto de líneas apropiados • Espacios apropiados
Pares {}	Asegurar que {} están adecuadamente casados
Operadores lógicos	• Verificar el uso apropiado de ==, =, //, y otros • Chequear que cada función lógica tenga su propio ()
Chequeo línea por línea	Chequear para cada LOC de código: • Sintaxis de la instrucción • Puntuación adecuada
Estándares	•Asegurase que el código esté acorde a l estándar de codificación
Abrir y Cerrar Fichero	Verificar que todos los ficheros estén: • Apropiadamente declarados • Abiertos • Cerrados

Tabla 8. Propuesta de la Lista de Chequeo de Código de PSP2

2.6.5 Lista de Chequeo de Diseño PSP2

Se propone utilizar la lista de chequeo de diseño pues el objetivo de la misma es guiar una revisión efectiva de diseño, se revisa el programa completo por categoría de lista de chequeo. Se asegura que los requerimientos, especificaciones y diseño de alto nivel estén completamente cubiertos por el diseño donde todas las salidas fueron producidas, todas las entradas necesarias fueron suministradas y todos los includes requeridos fueron colocados. Aquí se verifica que la secuencia del programa sea adecuada y que los tipos de datos abstractos como pilas listas y colas estén en el orden adecuado, además verifican que todos los lazos estén adecuadamente inicializados, incrementados y terminados, examinando cada sentencia condicional. Asegura que se opere bien con valores vacíos, llenos,

mínimos, máximos, negativos o cero para todas las variables. Protege las condiciones fuera de los límites y el desbordamiento de buffer y maneja todas las condiciones de entrada incorrectas. Esta lista de chequeo de diseño verifica que todos los datos de seguridad sean de fuente segura y al final revisa el diseño para ver si está conforme con todos los estándares de diseño aplicados. Una propuesta a la lista anteriormente descrita es la siguiente:

Nombre: _____ **Fecha:** _____

Nombre del Programa: _____**# Programa:** ____

Profesor: _____**Lenguaje:** _____

Objetivo	•Guía una revisión efectiva del diseño
General	•Revisar el programa completo para cada categoría de la lista de chequeo, una sola a la vez. •Una vez que se complete cada paso de la revisión, marcar cada elemento en el cuadro de la derecha •Completar la lista de chequeo para un programa o elemento de programa antes de comenzar a revisar el próximo.
Completo	Asegurarse que los requerimientos, especificaciones y diseño de alto nivel estén completamente cubiertos por el diseño: •Todas las salidas fueron producidas •Todas las entradas necesarias fueron suministradas •Todos los includes requeridos fueron colocados
Límites externos	•Cuando el diseño asume o cuenta con límites externos, determinar si el comportamiento es correcto con valores nominales, en los límites y más allá de estos
Lógica	Verificar que la secuencia del programa sea la adecuada: •Pilas, listas y otros estén en el orden adecuado •La recursión ocurra apropiadamente •Verificar que todos los lazos estén adecuadamente inicializados, incrementados y terminados •Examinar cada sentencia condicional y verificar todos los casos
Límites internos	•Cuando el diseño asume o cuenta con límite internos, , determinar si el comportamiento es correcto con valores nominales, en los límites y más allá de estos

Casos especiales	Chequear todos los casos especiales: •Asegurarse que operan bien con valores vacíos, llenos, mínimos, máximos, negativos o cero para todas las variables. •Proteger contra condiciones fuera de los límites y desbordamientos del búfer. •Asegurar que las condiciones imposibles son absolutamente imposibles •Manejar todas las condiciones de entradas incorrectas
Uso funcional	Verificar que todas las funciones, procedimientos u objetos son completamente comprendidos y adecuadamente usados. •Verificar que todas las referencias externas son definidas
Consideraciones del sistema	•Verificar que el sistema no causa que los límites sean excedidos •Verificar que todos los datos de seguridad son de fuentes seguras •Verificar que todas las condiciones de seguridad conforman las especificaciones de seguridad.
Nombres	Verificar lo siguiente: •Todos los nombres y tipos especiales están claros y están definidos específicamente •El alcance de todas las variables y parámetros está evidente y bien definido.

Tabla 9. Propuesta de la Lista de Chequeo de Diseño de PSP2

2.7 Guiones de PSP2.1

En PSP2.1 se adiciona en la fase de planeación un nuevo elemento, un intervalo de predicción de tamaño y tiempo. En la fase de desarrollo se documenta el diseño con las plantillas de diseño, se completa los formatos de especificación funcional, escenario operacional para registrar esta especificación, se produce un diseño para satisfacer la anterior especificación, se registra el diseño en los datos de especificación funcional, escenario operacional, especificación de estado y formatos de especificación lógica a medida que se van requiriendo.

Aparecen dos tareas nuevas: la revisión de diseño y la revisión de código, se recomienda para revisar el diseño seguir el guión de revisión de diseño y las listas de chequeo de diseño corrigiendo los defectos encontrados. Para la revisión del código se debe seguir el guión de revisión de código y las listas de chequeo de código, registrar los defectos en LOGD y registrar el tiempo en LOGT. Finalmente en la fase de postmortem deben estar completas las plantillas de diseño, las listas de

chequeo completas para las revisiones de diseño y código, un reporte de prueba y plan de proyecto completo.

2.8 Guiones de PSP3

En este nivel el programador debe ser capaz de desarrollar programas de hasta miles de LOC. Este proceso comienza con los requerimientos, luego se produce un diseño conceptual del programa entero donde se estima el tamaño y se planifica el desarrollo, a partir de este diseño de alto nivel se divide el producto y se divisa la estrategia cíclica. Cada ciclo se ejecuta con PSP2.1, se codifica, compila, se revisa y se prueba. Después de cada ciclo hay que reevaluar el plan y elaborar un reporte de ciclo indicando problemas y desviaciones.

En la fase de planeación se incorporan actividades como: producir los requerimientos y el plan de desarrollo, planes de tamaño, calidad recursos, cronogramas y producir un Log maestro de seguimiento de problemas. Posteriormente se realiza el Diseño de Alto Nivel (HLD), en el cual se produce el diseño y la estrategia de implementación, especificaciones funcionales, especificaciones de estado, escenarios operacionales, especificaciones de reutilización, estrategia de desarrollo y estrategia de pruebas. Después comienza la revisión de diseño de alto nivel, es recomendable anotar un problema sobresaliente en el Log de seguimiento de problemas y registrar en el Log de defectos todos los defectos encontrados. En la fase de postmortem se completa el formato resumen del ciclo con los datos reales del ciclo.

Se propone realizar una estrategia de desarrollo cíclico en la cual se subdivide el desarrollo del programa en módulos entre 100 y no más de 250 LOC nuevas y cambiadas, se localizan las LOC a ser desarrolladas en los ciclos y se registra el dato del tamaño en el resumen del ciclo.

2.8.1 LOG de seguimiento de problemas PSP3

Se propone utilizar el LOG de seguimiento de problemas pues proporciona una forma ordenada de registrar, seguir y administrar los problemas del proyecto. Para una buena utilización de este LOG de seguimiento de problemas hay que tener en cuenta el registro de los problemas que son diferidos para un manejo posterior y un registro de los problemas potenciales que pueden ser revisados y de los problemas que pueden haber sido muy revisados, priorizando estos problemas para ser atacados en las siguientes fases, siguiendo el manejo de estos problemas. Se numeran los problemas para evitar confusiones y se describe tan claro como sea posible teniendo en cuenta el cambio que se debe hacer, el problema que es revisado, la prueba que es realizada, la fase encontrada y así más adelante se

ejecuta la acción sugerida. Este registra la fecha en la cual se identificó y se registro el problema y describe como se resolvió el problema y la fecha en la cual se resolvió. El LOG se guarda en un lugar central para consultarlo al principio y durante la ejecución de cada fase de desarrollo. Una propuesta a lo anteriormente descrito es la siguiente:

Plan: _____ **Fecha:** _____

Estudiante: _____ **Real:** _____

Profesor: _____ **Lenguaje:** _____

# Problema	**Fecha:**	**Fase:**
Descripción:		
Resolución:		
Fecha:		
	Fecha:	**Fase:**
Descripción:		
Resolución:		
Fecha:		
	Fecha:	**Fase:**
Descripción:		
Resolución:		
Fecha:		

Tabla 10. Propuesta del LOG de Seguimiento de Problemas de PSP3

2.8.2 Formulario Resumen del Ciclo PSP3

Es recomendable utilizar el Formulario Resumen del Ciclo ya que mantiene los datos reales y estimados del ciclo de desarrollo en una forma conveniente y fácilmente recuperable. Un registro histórico de tales datos proporciona una base confiable que planea y hace seguimiento de proyectos grandes. Se mide y registra los valores durante o inmediatamente después de completar cada ciclo.

Los datos de defectos son reconstruidos rápidamente y el ciclo de la fase de removidos son fácilmente determinadas en el momento de remover, siendo la de la fase de infectados difícil de determinar. Una propuesta al formulario anteriormente descrito es la siguiente:

Plan: _____Fecha: _____

Estudiante: _____Real: _____

Profesor: _____Lenguaje: _____

Ciclos **Tamaño del Programa (LOC)**	**A la fecha**	1	2	3	4	5	Total
Base (B)							
Eliminados (D)							
Modificados (M)							
Adicionados (A)							
Reutilizados (R)							
Total nuevas y cambiadas(N)							
Total LOC (T)							
Total nuevas reutilizables							
Tiempo en fase (min.)							
Diseño							
Revisión de diseño							
Codificación							
Revisión de código							
Compilación							
Pruebas							
Total							
Defectos Inyectados							
Diseño							
Revisión de diseño							

Codificación							
Revisión de código							
Compilación							
Pruebas							
Total							
Defectos Removidos							
Diseño							
Revisión de diseño							
Codificación							
Revisión de código							
Compilación							
Pruebas							

Tabla 11. Propuesta del Formulario Resumen del Ciclo de PSP3

2.9 Formulario Resumen del Plan del Proyecto PSP3

El Formulario Resumen del Plan del Proyecto constituye el documento más importante del PSP, se viene elaborando desde PSP0 y se le van agregando elementos hasta llegar a elaborarse completo en PSP3.

Se recomienda realizar este documento correctamente para obtener resultados satisfactorios, demostrando un trabajo intenso, pero correctamente estructurado y organizado por parte del personal, un correcto cumplimiento sobre el uso de todos los documentos que plantea PSP, ya sean guiones, plantillas, formularios, listas de chequeo, contribuyen a un mejor posterior trabajo en equipo y por consiguiente los resultados finales del proyecto serán los esperados. A continuación se exponen ideas de cómo es más recomendable elaborar este resumen.

El objetivo de este formulario es mantener los datos estimados y reales del proyecto actual en una forma conveniente y fácil de recuperar. Se registran las LOC nuevas y modificadas planeadas por hora, se registra el tiempo planeado para el programa, se registran la suma de los tiempos planeados y tiempos reales para desarrollar todos los programas hasta la fecha, se registra el porcentaje de datos reutilizados planeados y reales, se registran los datos de los defectos planeados, reales y hasta la

fecha del porcentaje total del rendimiento, se calcula el costo de la calidad. Se estima el tamaño del programa en LOC antes del desarrollo.

El tiempo se debe dividir en etapas para registrar el tiempo planeado para cada fase, registrar el tiempo real para realizar cada fase. Después del desarrollo, se registra el total de defectos inyectados en cada fase. Para cada fase, se registra la suma que se lleva más los defectos inyectados en el último programa, se registra el valor del campo defectos inyectados. Después del desarrollo, registrar el total de defectos removidos en cada fase. Para cada fase, registrar la suma que se llevaba más los defectos removidos en el último programa. Para remover los defectos primero que nada se registran las eficiencias planeadas para este proyecto y las eficiencias obtenidas en este proyecto. Al final se registran las eficiencias reales para todos los proyectos hasta la fecha. Una propuesta al formulario anteriormente descrito es la siguiente:

Estudiante: _____ **# Fecha:** _____

Programa: _____ **# Programa:** _____

Profesor: _____**Lenguaje:** _____

Resumen	**Plan**	**Real**	**A la fecha**	
LOC/Hora				
Tiempo Planificado				
Tiempo Real				
CPI (Razón de Costo-Planificación)		*Plan/Real*		
% Reusadas				
% Nuevas Reusadas				
Defectos de Pruebas/KLOC				
DefectosTotales/KLOC				
Rendimiento				

Tamaño Prog (LOC)	**Plan**	**Real**	**A la fecha**	
Base (B)	*Medida*	*Medida*		

Eliminadas (E)	Estimada	Contada		
Modificadas (M)	Estimada	Contada		
Adicionadas (A)	N-M	T-B+D-R		
Reusadas ®	Estimada	Contada		
Total Nuevas-Modific (N)	Estimada	A+M		
Total LOC (T)	N+B-M-D+R	Medida		
Total Nuevas Reusadas				
LOC Objeto Estimado (E)				

Tiempo en fase (min.)	Plan	Real	A la fecha	% A la fecha
Planificación				
Diseño de Alto Nivel				
Revisión del Diseño de Alto Nivel				

Defectos inyectados	**Plan**	**Real**	**A la fecha**	**% A la fecha**
Planificación				
Diseño de Alto nivel				
Revisión del Diseño de Alto Nivel				
Diseño Detallado				
Revisión del Diseño detallado				
Codificación				
Revisión del Código				
Codificación				
Compilación				
Prueba				
Total Desarrollo				

Defectos eliminados	Plan	Real	A la fecha	% A la fecha
Planificación				
Diseño de Alto nivel				
Revisión del Diseño de Alto Nivel				
Diseño Detallado				
Revisión del Diseño detallado				
Codificación				
Revisión del Código				
Compilación				
Prueba				
Total Desarrollo				

Tabla 12. Propuesta del Formulario Resumen del Plan de Proyecto de PSP3

2.10 Conclusiones

Una vez hecho un análisis exhaustivo sobre los guiones, formularios, plantillas, métodos, planes, y listas de chequeo que plantea PSP en todos sus niveles, se puede llegar a la conclusión de la importancia que tiene su uso. PSP es un enfoque que muestra como aplicar métodos de Ingeniería de Software para el personal que desarrolla software en sus tareas diarias, con el objetivo de mejorar su trabajo.

PSP se basa en la definición de medidas y análisis de datos. De acuerdo a este análisis se verifica en que parte del proceso hay que ejecutar los cambios. Este proceso de evaluación y análisis se realiza en todas las facetas que marca PSP.

Uno de los puntos más importantes que se aplica con PSP es la reducción de defectos. Otro punto que es parte de este proceso es mejorar la planeación de los tiempos para la ejecución de cada una de las actividades.

CAPÍTULO 3

3.1 Introducción.

El presente capítulo pretende realizar una propuesta de aplicación de PSP por roles, con el objetivo de que cada integrante del proyecto aplique el PSP en dependencia del rol que desempeñe en el proyecto. Primero se hace un análisis sobre la estructura del proyecto, adecuando los roles que existen a los niveles, fases y formularios de PSP. Después se lanza la estrategia, la misma está estructurada en tres etapas, se realiza la distribución de los roles por sus fases y los artefactos que generan, así como los resultados esperados para esa etapa. Los artefactos que se generan son independientes de los generados normalmente en el proceso de la realización de la ingeniería de software.

3.2 Distribución de los roles (SACGIR).

Antes de lanzar la propuesta de solución de aplicación de PSP, se realiza un análisis sobre los roles, como están distribuidos y la cantidad de integrantes, con el objetivo de que cada integrante solo desarrolle de PSP lo que le corresponda en dependencia de su rol. Para la distribución de los roles fue necesario consultar “SACGIR Capital Humano”, documento en el cual se registra todo lo referente al capital humano, los integrantes del proyecto con sus roles, el año que cursan y las horas de trabajo a la semana. Se puede decir que existen 13 analistas (Analista de Gestión, Analista de Administración, Analista de Reportes, Analista de Control, Analista de Sistema o Principal), 6 diseñadores (Diseñador, 2 Diseñadores de Base de datos, Diseñador de Interfaz de Usuario, Diseñador de Prueba y un Diseñador Principal 3D), 22 implementadores, 2 arquitectos (Arquitecto Principal y Arquitecto Principal de Información), 2 planificadores, 2 integradores, 2 revisores técnicos, 2 probadores, 1 jefe de base de datos, 6 jefes de grupo de desarrollo, y 2 líderes de proyecto, quedando así la distribución de los roles se procede a realizar la vinculación con PSP.

3.3 Estrategia Propuesta:

3.3.3 Etapa I.

Esta primera etapa comienza con la aplicación de PSP por roles, en un primer módulo, de tamaño similar al tamaño de los módulos que desarrolla el equipo de trabajo normalmente. El objetivo de la misma es la distribución de los roles por las fases que estos desarrollan y los artefactos que estos generan, para obtener la primera generación de datos históricos y la estimación en tiempo y tamaño de este módulo.

Fases:

- Planificación: Se estima y planifica el trabajo.
- Diseño: Se diseña el programa.
- Codificación: Se implementa el diseño.
- Prueba: Se compila el programa, se repara y registran los defectos encontrados.
- Postmortem: Se registra el tiempo actual, defectos y tamaño de datos en el Resumen del Plan del Proyecto. Se actualiza el formulario PIP, con las deficiencias y las posibles mejoras encontradas en esta fase.

Artefactos generados:

- Cuaderno de Registro de Tiempo.
- Cuaderno de Registro de Defectos.
- Formulario Mejora de Proceso (PIP).
- Resumen del Plan del Proyecto.

Roles implicados:

- Analista de Sistema.
- Diseñador.
- Implementador.
- Probador.
- Líder de Proyecto.

Fase de Planificación:

En esta primera fase hay que hacer la estimación de recursos y el levantamiento de requisitos, el rol encargado de realizar estas tareas es el Analista de Sistema. Para guardar los tiempos de cada actividad de la planificación y de la captura de requisitos, se genera el artefacto, Cuaderno de Registro de Tiempo, con el objetivo de registrar el tiempo de realización de cada requerimiento. Para guardar los datos de los defectos cometidos en el levantamiento de requisitos se genera el artefacto, Cuaderno de Registros de Defectos. Los problemas encontrados, así como la propuesta para mejorar estos problemas, se guardan en el Formulario Mejora del Proceso, siendo otro artefacto generado.

Fase de Diseño:

En esta fase el diseñador realiza el diseño del programa y guarda los tiempos que demora en hacer el diseño, en el Cuaderno de Registro de Tiempo, con el objetivo de registrar el tiempo de realización del diseño del programa. Los defectos encontrados en este diseño se guardan en el Cuaderno de Registro de Defectos, se actualiza el Formulario Mejora del Proceso.

Fase de Codificación:

En esta fase se implementa el diseño, el implementador registra el tiempo que demora en implementar este diseño, en el Cuaderno de Registro de Tiempo, y los defectos encontrados en el Cuaderno de Registro de Defectos, actualiza los problemas y las mejoras en el Formulario Mejora de Proceso.

Fase de Prueba:

En esta fase se realizan las pruebas unitarias con métodos de caja blanca o negra, para encontrar posibles errores de implementación o de diseño, se guarda el tiempo, en el Cuaderno de Registro de Tiempo y se registran los defectos encontrados en el Cuaderno de Registro de Defectos. Los problemas encontrados, así como sus soluciones, se registran en el Formulario de Mejora del Proceso.

Fase de Postmortem:

En esta última fase, el líder del proyecto es el encargado de registrar todos los datos de tiempo y defectos, estos son utilizados para el análisis de la productividad de cada desarrollador y son registrados en el Resumen del Plan del Proyecto. Se actualiza el formulario PIP con las deficiencias analizadas en esta fase y sus posibles mejoras.

Resultados Obtenidos para esta primera etapa:

En esta primera etapa se obtiene como resultados, una primera generación de datos históricos, que pueden ser utilizados para la estimación de módulos similares, el Log de tiempos, de defectos y formulario PIP, de cada fase de desarrollo. Además se analizan los resultados de la productividad, con el objetivo de estimar en tiempo y tamaño para próximos módulos. El tiempo de desarrollo de este módulo, va a ser igual, al tiempo de desarrollo de cada una de las fases, más el tiempo en corregir los errores en la fase de prueba.

3.3.4 Etapa II.

Esta segunda etapa se sigue trabajando con la aplicación de PSP por roles, en un segundo módulo, de tamaño estándar. Se sigue con el mismo objetivo, la distribución de los roles por las fases que estos desarrollan y los artefactos que estos generan, para obtener la segunda generación de datos históricos y la estimación en tiempo y tamaño de este módulo, utilizando el método PROBE.

Fases:

- Planificación: Se planifica el trabajo y se documenta el plan.
- Diseño: Se diseña el programa.
- Revisión del diseño: Guía una revisión efectiva del diseño y se revisa el programa completo por categoría.
- Codificación: Se implementa el diseño.
- Revisión del código: Guía una revisión efectiva del código y verifica que se cubra todo el diseño.
- Prueba: Se compila el programa, se repara y registran los defectos encontrados.
- Postmortem: Se registra el tiempo actual, defectos y tamaño de datos en el plan.

Artefactos generados:

- Cuaderno de Registro de Tiempo.
- Cuaderno de Registro de Defectos.
- Formulario Mejora de Proceso (PIP).
- Lista de Chequeo de Diseño.

- Lista de Chequeo de Código.
- Resumen del Plan del Proyecto.

Roles implicados:

- Analista de Sistema.
- Diseñador.
- Implementador.
- Probador.
- Líder de Proyecto.
- Revisor Técnico.

Fase de Planificación:

En esta primera etapa la estimación está basada sobre los datos históricos del módulo anterior, a través del método PROBE. Aquí se lleva un formato de Cronograma y Tareas, en la herramienta Microsoft Project, por tanto no es necesaria la realización del Plan de Tareas y el Plan de Cronograma, propuesto por PSP. Se guardan los tiempos de cada una de las actividades en el Cuaderno de Registros de Tiempo, los defectos encontrados en el Cuaderno de Registro de Defectos, los problemas encontrados y las soluciones propuestas en el Formulario Mejora de Procesos.

Fase de Diseño:

En esta fase el diseñador realiza el diseño del programa y guarda los tiempos que demora en hacerse el diseño del programa, en el Cuaderno de Registro de Tiempo. Los defectos encontrados en este diseño se guardan en el Cuaderno de Registro de Defectos, se actualiza el Formulario Mejora del Proceso.

Fase de Revisión del Diseño:

En esta fase una vez hecho el diseño del programa, se procede a la revisión del diseño, realizada por el revisor técnico y se genera un nuevo artefacto, la Lista de Chequeo de Diseño. El tiempo demorado en realizar cada una de las actividades de la revisión del diseño, cada miembro del grupo de desarrollo lo registra en el Cuaderno de Registro de Tiempo. Los defectos encontrados en el Cuaderno

de Registros de Defectos. Los problemas encontrados, así como las soluciones propuestas, se registran en el Formulario Mejora de Proceso.

Fase de Codificación:

En esta fase se implementa el diseño, el implementador registra el tiempo que demora en implementar este diseño, en el Cuaderno de Registro de Tiempo, y los defectos encontrados en el Cuaderno de Registro de Defectos, actualiza los problemas y las mejoras en el Formulario Mejora de Proceso.

Fase de Revisión del Código:

En esta fase una vez implementado el diseño, se procede a la revisión del código, después de haberse revisado por el propio implementador, antes de la primera compilación y registrado en la Lista de Chequeo de Código. Se realiza otra revisión, por el revisor técnico, el cual actualiza la Lista de Chequeo de Código. El tiempo demorado en realizar cada una de las actividades de esta fase, cada miembro del grupo de desarrollo lo registra en el Cuaderno de Registro de Tiempo. Los defectos encontrados en el Cuaderno de Registros de Defectos. Los problemas encontrados, así como las soluciones propuestas, se registran en el Formulario Mejora de Proceso.

Fase de Prueba:

En esta fase se realizan las pruebas unitarias con métodos de caja blanca o negra, para encontrar posibles errores de implementación o de diseño, se guarda el tiempo, en el Cuaderno de Registro de Tiempo y se registran los defectos encontrados en el Cuaderno de Registro de Defectos. Los problemas encontrados, así como sus soluciones, se registran en el Formulario de Mejora del Proceso.

Fase de Postmortem:

En esta última fase, el líder del proyecto es el encargado de registrar todos los datos de tiempo y defectos, estos son utilizados para el análisis de la productividad de cada desarrollador y son registrados en el Resumen del Plan del Proyecto. Se actualiza el formulario PIP con las deficiencias analizadas en esta fase y sus posibles mejoras.

Resultados Obtenidos para esta segunda etapa:

En esta segunda etapa se obtiene como resultados, una precisa estimación basada en el método científico PROBE, recomendando el estudio de otros métodos de estimación de proyecto, ya que el método PROBE es bastante complejo. También se obtiene una segunda generación de datos históricos, que pueden ser utilizados para la estimación de módulos similares, el Log de tiempos, de defectos, el formulario PIP, las listas de chequeo de diseño y código, de cada fase de desarrollo.

Además se analizan los resultados de la productividad, con el objetivo de estimar en tiempo y tamaño para próximos módulos. El tiempo de desarrollo de este módulo, va a ser igual, al tiempo de desarrollo de cada una de las fases, más el tiempo en corregir los errores en la fase de prueba, más los tiempos de revisión de diseño y código.

3.3.5 Etapa III.

En esta tercera y última etapa se sigue la misma idea de la etapa dos, se sigue trabajando con la aplicación de PSP por roles, en un tercer módulo, de tamaño estándar, utilizando los datos históricos de módulos anteriores. Siguiendo los mismos procedimientos por fases y guardando la información de los datos en los distintos cuadernos de registros y formularios.

Al igual que en la etapa anterior, se obtiene como resultados, una precisa estimación basada en el método científico PROBE. También se obtiene una tercera generación de datos históricos, que pueden ser utilizados para la estimación de módulos similares, el Log de tiempos, de defectos, el formulario PIP, las listas de chequeo de diseño y código, de cada fase de desarrollo. Además se analizan los resultados de la productividad, con el objetivo de estimar en tiempo y tamaño para próximos módulos. El tiempo de desarrollo de este módulo, va a ser igual, al tiempo de desarrollo de cada una de las fases, más el tiempo en corregir los errores en la fase de prueba, más los tiempos de revisión de diseño y código.

La diferencia de esta etapa es la introducción del cálculo de las métricas de calidad en la fase de postmortem, para medir como esta la calidad del trabajo. Las métricas propuestas serían las siguientes: Índice de Calidad del Proceso y Tasa de Tiempo por Fase, de aquí el líder del proyecto las analiza según sus resultados, anota las deficiencias reflejadas en los resultados de las métricas, así como sus mejoras en el Formulario de Mejora del Proceso. Se recomienda seguir los pasos de la etapa tres para cada módulo asignado en un futuro.

3.4 Validación de la Estrategia Propuesta.

Una vez fundamentada la estrategia para aplicar el Proceso Personal de Software en los proyectos del Polo Productivo PetroSoft, se procede a la validación de la misma, con el objetivo de valorar y verificar que está a la altura de las necesidades y se puede aplicar, no sólo en el proyecto que se tomó como muestra en la investigación, sino también en otros similares.

Para la validación se seleccionaron dos especialistas que cuentan con una reconocida experiencia y prestigio profesional avalados por su alta calificación científico-técnica, conocimiento profundo de la Calidad de Software y de la aplicación del PSP en proyectos productivos.

Los especialistas en este caso son: Asesor de Calidad en la facultad 9: Ing. Ramsés Ibarrola Suárez y Miembro de Calidad UCI: Ing. Roig Calzadilla Díaz.

Opinión de los especialistas.

Ing. Ramsés Ibarrola Suárez.

Asesor de Calidad de la Facultad 9.

El presente trabajo tiene un buen nivel de calidad, además de que es de gran aporte científico. El trabajo es importante dado el problema que resuelve, tiene un buen nivel de comprensión y facilidad de uso y se adapta a diferentes entornos de la producción de software. La propuesta es útil y considero que debe aplicarse a todos los ámbitos.

Ing. Roig Calzadilla Díaz

Especialista Calidad UCI.

La estrategia propuesta en la investigación cuenta con los elementos necesarios para hacer menos complejo el uso de PSP en los proyectos productivos, la distribución por rol que plantea y la división en etapas ayuda al ingeniero a enmarcarse en la fase de PSP que le corresponde y por tanto, muy importante, ir desarrollándolo al mismo tiempo que realiza otros trabajos. Tiene la calidad y nivel científico requerido y debido a su utilidad y al problema que resuelve, por lo que se propone que debe ser aplicada en otros proyectos de producción de software.

3.5 Conclusiones.

Con la realización de este capítulo se da por concluida la propuesta de aplicación de PSP por roles, haciéndose de una forma más entendible y sencilla aplicando únicamente los niveles y formularios de salida necesarios para registrar y controlar el proceso de software personal en la construcción de un producto informático. En estas tres etapas propuestas se trabaja rigurosamente siguiendo las fases del PSP, desglosando cada tarea en las actividades que servirán como guía para el trabajo, empleando la estrategia planteada.

CONCLUSIONES

El gran desarrollo alcanzado por la industria de software y la necesidad de productos confiables, precisos y funcionales, hace que aumente cada vez más la exigencia por parte de clientes y usuarios. Dada la importancia de obtener una mejora continua en productos, procesos y servicios, en este documento se presentan una estrategia de cómo aplicar el PSP de forma disciplinada y ordenada para estructurar el trabajo personal del ingeniero, constituyendo una labor determinante a la hora de planificar , medir y gestionar dicho trabajo.

De esta forma se da por concluida la presente investigación, dejando por sentado el cumplimiento de cada objetivo propuesto y poniendo a consideración de los líderes de proyectos del polo productivo de PetroSoft el empleo de la estrategia de planificación propuesta en el desarrollo de los productos. Dicha estrategia garantiza al proyecto en construcción:

- Puntualidad y calidad en la entrega de cada elemento perteneciente al producto en desarrollo o concluido el mismo.
- Una planificación real y bien estimada del tiempo y los recursos empleados en el proceso.
- Mejor aprovechamiento de los recursos materiales y del capital humano destinado al proyecto.
- Alto nivel de organización de los planes de trabajo y la documentación referente al expediente de proyecto propuesto por Calidad UCI.
- Desarrollo ágil y eficiente del proceso de construcción del software en cuestión.

Proporciona métodos detallados de planificación de tiempos y estimación de actividades. Muestra a los Ingenieros como controlar su rendimiento frente a estos planes y explica como los procesos definidos guían su trabajo. PSP ha sido introducido en muchas organizaciones y actividades del proceso de desarrollo de software y en la calidad de los productos que se obtienen. Puede considerarse como una disciplina que proporciona un marco de trabajo estructurado para desarrollar habilidades personales.

RECOMENDACIONES

Luego de haber hecho un estudio riguroso del Proceso Personal de Software y conociendo la importancia que presenta para el trabajo personal de los desarrolladores se recomienda:

- Aplicar la estrategia propuesta al proyecto SACGIR.
- Adaptar esta propuesta a otros proyectos, para demostrar su capacidad de planificación y gestión de proyectos.
- Que el documento quede como fuente de consulta y bibliografía en este tema de PSP.
- Tener presente el uso de datos históricos para el desarrollo de otros módulos.
- Estudiar otro método de estimación menos complejo que el PROBE.
- Que los desarrolladores tomen conciencia de cuan importante es el uso correcto del PSP para el desarrollo de los proyectos.

REFERENCIAS BIBLIOGRÁFICAS

[1]. *Humphrey, W., "Introducción al Proceso de Software Personal". 2001.*

[2]. *Humphrey, W., "A self improvement Process for Software Engineers". 2005.*

[3]. *Pressman, R.S., "Ingeniería de Software. Un enfoque práctico", Capítulo 1. 2005.*

[4]. *Disponible en: http://www.ingenierosoftware.com/calidad/cmm-cmmi.php.*

[5]. *Pressman, R.S, "Ingeniería de Software: un enfoque práctico", 5ta. Edición.*

[6]. *Disponible en:http://www.usm.edu.ec/abedini/spice/spice4.htm.*

[7]. *Disponible en: http://hornet.ls.fi.upm.es/DSP/Lecciones*

[8]. *Humphrey, W., "PSP. A Self – Improvement Process for Software Engineers", Capítulos 8, 9, 10, 11 y 12. 2005.*

[9]. *Humphrey, W., "A self improvement Process for Software Engineers", Capítulos 13 y 14. 2005.*

[10]. *Disponible en: http://html.rincondelvago.com/prueba-de-software.html.*

[11]. *Myers, G.J., "The Art of Software Testing". 1979.*

[12]. *IEEE STD 1074-1997, IEEE Standard for Developing Software Life Cycle Processes: Institute of Electronics Engineers, 1997. .*

[13]. *Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El Proceso Unificado de Desarrollo de Software. La Habana: Félix Varela, 2004. pág. 281. Vol. 1.*

[14]. *Ingeniería del Software, Un Enfoque Práctico. Holguín: Empresa Poligráfica José Miró Argenter, 2005. págs. 294-298. Vol. 1.*

[15]. *Ingeniería del Software, Un Enfoque Práctico. Holguín: Empresa Poligráfica José Miró Argenter, 2005. págs. 286-293. Vol. 1*

[16]. *Humphrey, W., "A self improvement Process for Software Engineers", Capítulos 1 y 2. 2005.*

[17]. *Humphrey, W.S., "PSP A Self-Improvement Process for Software Engineers", Capítulos 5 y 6. 2005.*

[18]. *Humphrey, W., "PSP. A Self Improvement Process for Software Engineers", Capítulo 3. 2005.*

BIBLIOGRAFÍA

1. Pressman, Roger S. Ingeniería del software. Un enfoque práctico.

2. Calidad en ingeniería del software, disponible en: http://*dmi.uib.es/~bbuades/calidad/index.htm*

3. Un enfoque actual sobre la calidad del software, Oscar M. Fernández Carrasco, Delba García León. Alfa Beltrán Benavides, disponible en: http://bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.htm.

4. Calidad del software: gestión eficaz de los procesos de desarrollo de software para satisfacer la calidad, disponible en: http://www.fundacion.unavarra.es/CursosdeVerano/Calidad_del_Software.pdf

5. Control de la calidad del software, disponible en: http://www.sdl.com/es/services/services-sdl-languageservices/services-sdl-software-qa.htm

6. Modelos de calidad de software y software libre, Ernesto Quiñones, disponible en:

http://www.eqsoft.net/presentas/modelos_de_calidad_y_software_libre.pdf

7. Estimación del coste de la calidad del software a través de la simulación del proceso de desarrollo, disponible en:http://www.unab.edu.co/editorialunab/revistas/rcc/pdfs/r21_art6_c.pdf.

8. Pruebas de software, disponible en: http://lsi.ugr.es/~ig1/docis/pruso.pdf

9. Pruebas de software, disponible en: http://www.greensqa.com/archivos/

10. Pruebas de software, disponible en: http://lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt

11. Pruebas del software, disponible en: http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf

12. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El Proceso Unificado de Desarrollo de Software.

13. Bedini G, Alejandro. Calidad Tradicional y de Software.

14. Straub, Pablo. El costo de la calidad. 2006.

15. Humphrey, W., "Introducción al Proceso de Software Personal". 2001.

16. Humphrey, W., "A self improvement Process for Software Engineers". 2005.

GLOSARIO DE TÉRMINOS

Calidad: Calidad de software. Satisfacción de las necesidades de los usuarios.

Aseguramiento de la Calidad: Conjunto de acciones planificadas y sistemáticas que son necesarias para proporcionar la confianza de que un producto o servicio satisface los requisitos de calidad preestablecidos.

Control de la Calidad: Técnicas y actividades de carácter operativo utilizadas para satisfacer los requisitos de calidad.

IEEE: (Institute of Electrical and Electronics Engineers). Asociación de profesionales norteamericanos que aporta criterios de estandarización de dispositivos eléctricos y electrónicos.

ISO: Serie de procedimientos y directrices que le permiten homogenizar lenguajes y bases técnicas a nivel mundial, con el fin de seleccionar y mejorar procesos.

CMMI: Modelo de Madurez de Capacidades Integrado, es un modelo de calidad del software que clasifica las empresas en niveles de madurez.

SPICE: Software Mejora de Procesos y Capacidad de Determinación. Es un emergente estándar internacional de evaluación y determinación de la capacidad y mejora continua de procesos de ingeniería del software.

Software: Software es un término genérico que designa al conjunto de programas de distinto tipo (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador.

Metodología: Es un conjunto de procedimientos, técnicas, instrumentos y documentos que ayudan a los analistas y programadores en sus esfuerzos para obtener un nuevo sistema informativo.

Proceso: Secuencia de actividades invocadas para producir un producto de software.

Métricas: Evaluación a través de los requerimientos de tiempo y espacio del programa. Están ligadas tanto al algoritmo como a la arquitectura paralela de destino. Medición de la calidad.

Fase: Son los pasos en que se descomponen las metodologías. Cada fase puede o no estar subordinada a otra fase, pudiendo existir entre ellas relaciones de dependencia.

PSP: Proceso Personal de Software.

Artefacto: Pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar las actividades; representa un área de responsabilidad y es candidata a ser tenida en cuenta para el control de la configuración. Un artefacto puede ser un modelo, un elemento de un modelo, o un documento.

Modelo: Cosa que ha de servir de objeto de imitación. Objeto, construcción u otra cosa con un diseño del que se reproduce más iguales. Esquema teórico de un sistema o de una realidad compleja que se elabora para facilitar su comprensión y estudio.

Rol: Papel, cometido o función que tiene o desempeña que interpreta un actor.

Desarrollador: Persona que trabaja directamente en el desarrollo de un proyecto de software, dígase: analista, programador, diseñador, etc.

Grupo de desarrollo: Dos personas o más que trabajan para lograr una meta, objetivo o misión común, donde cada individuo tiene asignado un rol específico y donde el completamiento de la misión depende de los miembros del equipo.

Defecto: Un proceso, una definición de datos o un paso de procesamiento incorrectos en un programa.

Prueba: Prueba de software. Ejecución de un sistema bajo condiciones específicas, se observan y se analizan los resultados realizándose una evaluación de los mimos.

PROBE: Método de estimación basado en Proxy, el cual determina el valor estimado de las líneas de código de un programa y tiempo que se tomó para ello.