IMPLEMENTACIÓN DE UN SISTEMA INTEGRADO UTILIZANDO PROCESOS DE SOFTWARE EN EQUIPO (TSP)

José Luís Asencio Mera[1], Carlos Mauricio Echeverría Goyes[2], Cynthia Denisse Echeverría Goyes [3], Mónica Villavicencio[4]

[1] Ingeniero en Computación 2006
[2] Ingeniero en Computación 2006
[3] Ingeniera en Computación 2006
[4] Directora de Tesis, Ingeniera en Computación, Escuela Superior Politécnica del Litoral.

Abstract

This article summarizes the experiences of a team of three software developers who were assigned a task oriented to apply "Team Software Process" TSP. Their roles were: Planning Administrator, Development Administrator and Configuration Administrator. The main objective of this task was to instill in them awareness of the benefits of using TSP. To do so, a development environment was elaborated that consisted of implementing an integrated system that included the following four modules: production costs, by project budget work orders, and control of inventory and warehouse. During the process, metrics, standards, work timetables, and so on were used in order to helped them to improve the quality of software and to enhance the team productivity.

Resumen

Este artículo resume las experiencias de tres de cinco desarrolladores de software que conformaron un equipo experimental para la aplicación de un “Proceso de Software para equipos” TSP. Los roles desempeñados por ellos fueron: administrador de planificación, administrador de desarrollo y administrador de configuración. El objetivo de este trabajo fue conocer los beneficios de emplear TSP. Para el efecto, se elaboró un ambiente de desarrollo, el cual consistió en implementar un sistema integrado que contiene los módulos de costos de producción, presupuesto por obra, órdenes de trabajo y control de inventario y bodega. Durante el proceso se utilizaron métricas, estándares, cronogramas de trabajo, etc. que nos ayudaron a mejorar la calidad del software y mejorar la productividad del equipo.

Introducción

Ecuador, al igual que otros países de América Latina, ha decidido apostarle al desarrollo de software, actividad que ha crecido en importancia durante los últimos años. Sin embargo, su apogeo no garantiza una buena perspectiva para éste sector ya que existen inconvenientes en el desarrollo de los sistemas de software, como son: grandes retrasos en la programación, inconsistencias en su funcionalidad, planificación irreal, etc.; lo cual redunda en la calidad del producto entregado.

Según el análisis exploratorio realizado en el marco del proyecto VLIR, se determinó que de 77 empresas, pocas han utilizado, utilizan, o piensan utilizar estándares de calidad para el desarrollo de software (1). Ante la crítica situación de las empresas desarrolladoras de software, se ejecutó el presente proyecto de tesis para evaluar capacidades de la ingeniería de software, a nivel individual y de equipo, mediante la utilización del modelo TSP (Team Software Process) propuesto por el Instituto de Ingeniería de Software SEI (1).

En este proyecto de tesis se emplearon estándares, plantillas y métricas que sirvieron para evaluar el desempeño del grupo y la calidad del sistema, enfocadas en tres roles principales: administrador de desarrollo, administrador de planificación y administrador de configuración.

El TSP y su aplicación en el proyecto

El TSP es un modelo de trabajo en equipo enfocado a aminorar varios de los problemas, tanto técnicos como administrativos, que se presentan en el desarrollo de software (2). El TSP provee un esquema de trabajo donde cada desarrollador tiene bien definido sus roles, sus actividades, y sus responsabilidades. Además, el TSP incluye procedimientos para la mejora continua del proceso de desarrollo, la mejora de la calidad del software producido, la mejora de la estimación del tiempo de desarrollo, la disminución de defectos en el producto y la promoción de la integración del equipo de desarrollo (2). Es decir, el TSP apoya tanto al equipo de desarrollo como a los administradores del proyecto para la culminación a tiempo y dentro del presupuesto de proyectos de desarrollo de software (2). Adicionalmente, el TSP muestra como aplicar los conocimientos de ingeniería de software y los procesos principales en el ambiente de trabajo en equipo, ayudando también a que el equipo gane experiencia en la planificación y administración de proyectos de software (3).

Para cumplir los objetivos que plantea TSP, se necesita que cada miembro del equipo entienda las virtudes y carencias de los otros miembros, que los apoye y que esté dispuesto a pedir ayuda cuando se requiera. Trabajar en equipo no es una habilidad que se adquiere al nacer, se adquiere a través de la práctica y se mejora día a día con la experiencia (2). Normalmente, los ingenieros de software desarrollan productos a partir de sus propios métodos y técnicas, o a partir de ejemplos obtenidos de las mejores prácticas.

Al equipo de trabajo en este proyecto, PSP y TSP permitió controlar individualmente y en grupo el desempeño en el desarrollo del presente proyecto de tesis. Un equipo de trabajo que utiliza TSP está conformado por 5 personas, las mismas que durante todos los ciclos del proyecto asumen los siguientes roles: 1)Líder de Equipo, 2)Administrador de Calidad, 3)Administrador de Desarrollo, 4)Administrador de Planificación y 5) Administrador de Configuración. Este artículo se enfoca en los 3 últimos roles.

El propósito de esta investigación fue obtener y validar esta disciplina de desarrollo propuesta por el SEI, utilizando estándares, recolectando y analizando métricas que ayuden en la mejora de los procesos para demostrar así la aplicabilidad del TSP en este proyecto en particular. Esta investigación se enfocó en obtener un producto de software que cumpliera con los procedimientos establecidos por TSP, tomando en consideración los requerimientos de una empresa para automatizar sus procesos del negocio. El producto desarrollado fue un “Sistema Integrado de Control de Costos de Producción, Órdenes de Trabajos, Presupuestos por Obra, Bodega y Control de Inventario”, que apoye a la toma de decisiones de los directivos y funcionarios de la empresa, quienes permitieron realizar esta investigación.

Roles de administrador de desarrollo, planificación y configuración.

Este artículo se enfocó principalmente en tres roles en los cuales se describen los objetivos y responsabilidades de cada uno de ellos (ver Tabla 1).

Roles	Objetivo	Responsabilidades
Administrador de Desarrollo	Conducir al equipo en la definición, diseño, desarrollo y pruebas del producto.	Dirigir el equipo en la implementación del proyecto. Dar soporte en el desarrollo del proyecto. Verificar código fuente según los estándares definidos. Conducir al equipo en la generación de la documentación técnica del proyecto.
Administrador de	Apoyar y guiar a los integrantes del equipo en la planificación y	Desarrollar y mantener el programa de trabajo actualizado.

Planificación	seguimiento de su trabajo.	Verificar el cumplimiento de las actividades programadas. Controlar el registro de horas individuales. Comparar el progreso del equipo con lo planeado.
Administrador de Configuración	Colaborar con el equipo en la determinación, obtención y mantenimiento de las herramientas necesarias para cumplir con las necesidades administrativas y aplicar la tecnología definida.	Controlar los cambios que afectan a los elementos de configuración. Definir herramientas de desarrollo y de control de versiones. Mantener el interés del equipo en el uso de las herramientas. Evaluar las solicitudes de cambios. Mantener el sistema de administración de riesgos.

Tabla No. 1
Descripción de Objetivos y Responsabilidades por Rol

Para la asignación de roles, se tomó en cuenta las características más relevantes de cada miembro del equipo, como son: tener conocimiento de los métodos de diseño, tener gusto por construir cosas, no ser resistente al cambio, seguir un esquema de trabajo definido, tener conocimiento sobre las herramientas y sistema de apoyo.

Plantillas, modelos, metodologías y estándares

La Tabla No.2 presenta las plantillas, y estándares aplicados por cada administrador.

ROL	PLANTILLA	ESTANDARES Y PLANES DE APOYO
Administrador de desarrollo	Registro Soporte de Desarrollo Registro Longitud de Código	Estándar de Programación
Administrador de planificación	Task (Registro de Tareas)	Planificación del trabajo
Administrador de configuración	CCR (Registro de Solicitud de cambio)	Plan de Gestión de configuración

Tabla No. 2
Plantillas y Estándares por cada rol

El propósito de registrar la longitud de código en el proyecto fue saber el tamaño de los módulos que se iban generando. Los elementos que se tomaron en consideración para registrar la longitud, fueron las clases, módulos y procedimientos almacenados de la base de datos. Al igual que el registro de longitud de código, también se registró el número de requerimientos de soporte a los miembros del equipo, el cual ayudó a identificar los errores más frecuentes que se presentaron a medida que se desarrollaba el proyecto.

La plantilla “Task” tiene el propósito de estimar el tiempo de desarrollo para cada tarea del proyecto. En esta plantilla se toma en consideración los siguientes aspectos: el responsable del equipo de trabajo, fecha de registro de las tareas con su respectiva distribución del personal, la planificación en horas estimadas y el tamaño real. En el presente caso, la plantilla Task fue utilizada por cada incremento.

Con la plantilla “CCR” se registraron todas las solicitudes de cambios, los cuales fueron sometidos a una evaluación para determinar su impacto. Todos estos procesos están contemplados en el plan de gestión de configuración.

Metodología de Desarrollo

El modelo de desarrollo incremental fue utilizado para desarrollar el producto de software. Este modelo es iterativo, ya que para obtener el producto final se deben implementar e integrar cada uno de los incrementos (4). En un proceso de desarrollo incremental, se identifican todos los

servicios de acuerdo a su importancia. Posteriormente, se definen varios incrementos, en donde cada uno proporciona un subconjunto de funcionalidad del sistema. La asignación de servicios a los incrementos depende de la prioridad del servicio. Los servicios de prioridad más alta son los que se entregan primero al cliente (4). Cada incremento consta de cuatro fases: análisis, diseño, implementación, y pruebas. Posteriormente, en este artículo mostraremos las etapas por cada incremento.

En esta investigación, se comprobó que los servicios que sufrieron más pruebas fueron los primeros, es decir los de mayor prioridad. Además, del primer incremento obtuvimos experiencia para poder desarrollar los siguientes.

Etapas del Proyecto

El proyecto fue dividido en tres etapas básicas para cada incremento: definición, desarrollo y producción (5), tal como se muestra en la figura 1.

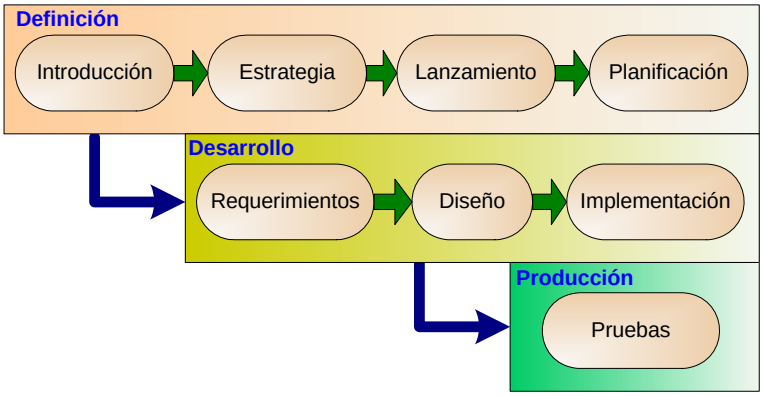


Figura No. 1
Divisiones de las etapas básicas del proyecto

La fase a la que el equipo de desarrollo le dedicó mas tiempo fue la de requerimientos. Algunas de las razones se listan a continuación:

- El cliente no tenía bien definidos sus procesos de negocios.
- La frecuencia de cambio de los requerimientos del negocio fue alta.
- No hubo una definición formal de los requerimientos al inicio del proceso de desarrollo.
- No se definió un alcance de los módulos del sistema.

Otra de las fases que requirió mayor tiempo fue la de pruebas, debido a la magnitud de unidades a probar y verificar. También, se tomaron en cuenta las pruebas de regresión en los módulos que sufrieron cambios.

El sistema integrado cuenta con 8 módulos:

MCIB Módulo de Control de Inventario y Bodega.
MNO Módulo de Nómina.
MOC Módulo de Compras.
MOT Módulo de Órdenes de Trabajo.
MFAC Módulo de Facturación.
MPO Módulo de Presupuesto por Obra.
MCP Módulo de Costos de Producción.

Métricas utilizadas en el proceso de desarrollo de software

Las métricas nos proveen de mediciones para poder analizar el desempeño de los procesos ejecutados por el equipo de trabajo. La Tabla No.3 presenta las métricas recopiladas y los responsables de su control y seguimiento.

RESPONSABLE	MÉTRICA	DESCRIPCION	JUSTIFICACION
Administrador de Desarrollo	Longitud de código por tipo de fuente y por incremento.	Total de código fuente categorizado desarrollado en un incremento.	Nos ayuda a medir el nivel de complejidad de los sistemas con respecto a las líneas de código.
	Número de veces que se dio soporte a los demás miembros del equipo.	Total de soporte a los miembros del equipo en las etapas de diseño.	Nos ayuda a la identificación de los principales problemas y en definir acciones que pueden ser tomadas para resolverlos.
	Componentes reutilizados en los diferentes incrementos.	Cantidad de elementos reutilizados que se tomaron en cuenta en el desarrollo del proyecto.	Nos ayuda en la identificación de componentes reutilizables en los incrementos que se ejecutaron.
Administrador de Planificación	Horas de equipo trabajadas por incremento.	Total de horas que trabajo el equipo por cada incremento tomando en consideración las etapas definidas en el proyecto.	Nos ayuda a realizar un seguimiento sobre el tiempo dedicado por equipo a las tareas definidas en cada incremento.
	Comparativo de horas planificadas por rol vs. Horas trabajadas.	Total de horas trabajadas por cada rol con respecto a las horas planificadas.	Nos ayuda a evaluar si el equipo pudo cumplir con las tareas programadas.
Administrador de Configuración	Número de cambios de requerimientos por módulo.	Total de cambios registrados en el desarrollo de cada módulo.	Nos ayuda a verificar la frecuencia de cambios que se obtuvieron en el desarrollo de cada módulo.
	Eficiencia en realizar los cambios.	Comparación entre tiempos dedicados a realizar cambios entre diferentes incrementos.	Nos ayuda a verificar si el proceso de cambio surte efecto en el desarrollo de software.
	Número de versiones de elementos de configuración.	Total de versiones que se obtienen para los elementos de configuración.	Nos ayuda a identificar los cambios efectuados desde el punto de vista de versiones.

Tabla No. 3
Métricas utilizadas en el desarrollo del software

Longitud de código.- A través del total de líneas de código (LOC) se puede determinar si un sistema es complejo o no. A mayor número de líneas de código más complejo es el sistema (10).

En la Tabla No. 4, se presentan las líneas de código para cada módulo. El cálculo de LOCs incluye procedimientos, funciones, variables, clases, formularios, comentarios. Como se puede apreciar, los módulos más extensos fueron los de Inventario y Nómina. Coincidentemente, éstos son los que presentaron mayor dificultad durante el desarrollo del sistema debido a la particularidad del negocio.

TIPO DE FUENTE	MCIB	MCP	MOT	MPO	MNO	MFAC	MOC	TOTAL GENERAL
Clases	2806	774	1790	1688	1633	981	877	10549
Módulos	273	273	273	273	273	273	273	1911
Pantallas	28800	3970	9795	12470	7215	5378	3900	71528
Stored procedures	6156	961	1465	1630	5828	1176	1311	18527
TOTAL GENERAL	38035	5978	13323	16061	14949	7808	6361	102515

Tabla No. 4
Locs por Módulos Desarrollados

Porcentaje de **Reutilización de código.-** La reutilización de código contribuye a mejorar la calidad del software debido al uso de componentes previamente probados (11).

Nuestro equipo de trabajo fomentó la reutilización de código fuente durante el desarrollo de los módulos, esto permitió disminuir proporcionalmente los tiempos de programación. Los módulos en donde hubo mayor porcentaje de reutilización de código fuente fueron en los módulos de inventario y presupuesto por obra como se muestra a continuación:

MÓDULO	LOC
MCIB	255590
MCP	45033
MOT	89613
MPO	129747
MNO	73190
MOC	33258
MFAC	40850
TOTAL LOCS REUSADAS	**667281**

Tabla No. 5
Locs Reutilizados por Módulo

Las clases, stored procedures, funciones y procedimientos fueron los elementos que se tomaron en consideración para identificar los componentes reutilizables.

Número de horas trabajadas.- La Tabla No. 6 muestra el número de horas trabajadas versus las planificadas, pudiéndose apreciar una notable mejoría en la estimación de horas. Los datos de esta métrica se recopilaron en la etapa de desarrollo usando las plantillas Task, Logt y el plan del equipo. El tiempo de duración del proyecto de tesis fue de 52 semanas en donde algunos riesgos fueron identificados, algunas tareas fueron desestimadas y los requisitos cambiaron y crecieron. Para simplificar la historia de este proyecto, se mostrarán los datos obtenidos en las semanas 20, 28, 36 y 52.

	PLANIFICADO	TRABAJADO
Datos Semana 20	646.8	1006.15
Datos Semana 28	373.7	461.8
Datos Semana 36	118.9	124.4
Datos Semana 52	342.0	353.9

Tabla No. 6

Horas Planificadas vs. Horas Trabajadas

Como se puede apreciar, existe una notable mejoría en la estimación de horas debido fundamentalmente a que el equipo de trabajo adquirió experiencia en el modelo del negocio y en la aplicación del TSP. Adicionalmente, se aprendió a conocer el comportamiento de los usuarios.

Horas planificadas por rol vs. Horas trabajadas.- Esta métrica permite medir el nivel de cumplimiento de trabajo de cada miembro del equipo con respecto a lo planificado. Para obtener la métrica, se tomó en consideración las horas trabajadas por cada administrador. Los factores que afectaron a esta métrica fueron: el uso de una nueva metodología de desarrollo, la falta de experiencia entre los administradores y la experiencia adquirida por el equipo en el proceso de desarrollo.

ROL	HORAS PLANIFICADAS	HORAS TRABAJADAS	PORCENTAJE DE DESFASE
Líder Equipo	391.3	538.1	37.5%
Adm. Desarrollo	375	509	35.7%
Adm. Planificación	261.8	314.1	20.1%
Adm. Calidad	256.2	312.2	21.8
Adm. Configuración	312.8	364.55	17.44%

Tabla No. 7
Comparativa de Horas Planificadas por rol vs. Horas Trabajadas con su porcentaje de desfase

En la Tabla No. 7, vemos los desfases en horas de los roles, el desfase se encuentra en el rango del 17% al 37.5%. Estos desfases se debieron a los cambios constantes de requerimientos de módulos a su cargo, a la falta de experiencia en la herramienta de desarrollo, a la falta de experiencia en planificar y al uso de una nueva metodología.

Número de cambios por módulo.- A través de esta métrica llevamos el control de cambios por cada módulo. En la tabla 8 se visualiza la reducción de cambios por módulo, esto se debe fundamentalmente a que el equipo de desarrollo adquirió más conocimiento del dominio del negocio. Adicionalmente, se establecieron políticas para administrar los cambios.

MÓDULOS	NÚMERO DE CAMBIOS
MCIB	10
MCP	4
MFAC	7
MNO	11
MOC	6
MOT	7
MPO	7

Tabla No. 8
Número de cambios por Módulo desarrollado

Eficiencia en realizar cambios.-Para esta métrica, se toma en consideración los tiempos de evaluación de la solicitud de cambio, el tiempo de implementación del cambio y los tiempos de revisión del cambio que se registraron el las pruebas unitarias.

PROCESOS DE CAMBIOS	MCIB	MCP	MFAC	MNO	MOC	MOT	MPO	TOTAL GENERAL
EVALUACION	4.2	2.6	3.9	4.6	3.6	3.4	2.9	**25.2**
IMPLEMENTACION	5.3	3.5	5	5.7	4.8	4.8	3.4	**32.5**
REVISION	0.2	0.2	0.3	0.1	0.5	0.15	0.2	**1.65**
TOTAL GENERAL EN HORAS	**9.7**	**6.3**	**9.2**	**10.4**	**8.9**	**8.35**	**6.5**	**59.35**

Tabla No. 9
Horas Trabajadas en pruebas por Módulo

Como se puede observar, los tiempos para realizar pruebas de los primeros módulos son altos comparados con los módulos finales. El orden en el cual los módulos fueron desarrollados fue el siguiente:

- Módulo de Control de Inventario y Bodega, Nómina.
- Módulo de Compras, Órdenes de Trabajo.
- Módulo Presupuesto por Obra, Facturación.
- Módulo de Costos de Producción.

Realizando una comparación entre los tiempos que se registraron para un solicitud de cambio al inicio del desarrollo del software y al final del mismo, nos damos cuenta que el equipo pudo reducir el tiempo de los procesos de cambios, hubo una disminución considerable en la etapa de evaluación y la etapa de implementación del cambio, por el contrario, en la etapa de revisión del cambio el tiempo se mantuvo estable, con esta medición podemos definir que aumentó el grado de eficiencia (12).

Número de versiones de los elementos de configuración.- Nos permite identificar el cambio de versiones de los elementos de configuración previamente definidos. A medida que el equipo ganaba experiencia en TSP se observó una disminución en el número de versiones. Para esta métrica se identificaron los elementos de configuración definidos en las líneas base que se encuentran en el plan de configuración.

En el proyecto se definieron 3 líneas base, la primera es la línea base de definición el cual agrupa todos los elementos de configuración que se encuentran en las fases de introducción, estrategia, lanzamiento y planificación como por ejemplo: objetivos del equipo, objetivo del producto, definición de la arquitectura del proyecto, definición de estándares y la planificación del proyecto. En la segunda línea base a la cual se la llamo de desarrollo, se encuentra los elementos de configuración de las fases de requerimientos, diseño e implementación como por ejemplo: requerimientos del cliente, requerimiento del desarrollador, diseños detallados y los módulos desarrollados. Para la tercera línea base definida como línea base de producción, se tomó en consideración los elementos de configuración de la fase de pruebas e implantación de los módulos como por ejemplo: pruebas unitarias, pruebas de regresión, producto final integrado.

Para una mejor evaluación de la métrica tomamos en consideración las etapas de desarrollo y producción, se tomó en cuenta los siguientes elementos de configuración: Requerimientos del cliente, Requerimientos del desarrollador, Diseño Detallado

Para la evaluación de la métrica en la etapa de producción, se tomó en cuenta las pruebas: Unitarias, de Aceptación y de Regresión.

En la Tabla No. 10 se muestra el total de número de versiones obtenidas por módulo y por etapa. Si observamos el cuadro comparativo de esta tabla, podemos ver que existe una disminución en el número de versiones de los elementos de configuración que se encuentran en la etapa de desarrollo como por ejemplo el MCIB se obtuvo un total de 17 versiones de los elementos de configuración, y en MCP que fue uno de los últimos módulos se obtuvieron un total de 6 versiones.

	MCIB	MCP	MFAC	MNO	MOC	MOT	MPO	TOTAL GENERAL
DESARROLLO	17	6	9	14	8	12	8	**74**
PRODUCCION	8	4	8	8	8	5	5	**46**
TOTAL GENERAL	**25**	**10**	**17**	**22**	**16**	**17**	**13**	**120**

Tabla No. 10
Versiones obtenidas en desarrollo y producción

De igual manera en la etapa de producción las versiones que se obtuvieron con los módulos finales son menores en referencia a los módulos iniciales.

Análisis Comparativo: Antes y Después del Uso del TSP

Antes de adquirir conocimiento acerca de la metodología TSP ningún miembro del equipo de trabajo tenía como procedimiento utilizar estándares tanto de documentación como de desarrollo. No efectuaban planificaciones; por lo tanto, no se distribuían las tareas de forma efectiva cuando se trabajaba en grupo. Rara vez documentaban el software y como consecuencia de esto, el análisis y diseño de las aplicaciones desarrolladas era muy pobre. No definían fechas de entrega ni respetaban un cronograma de trabajo. No tenían como costumbre efectuar pruebas cuando se efectuaban desarrollos, ni realizaban inspecciones en la documentación. Tampoco controlaban los cambios efectuados en el sistema. En suma, el grupo de trabajo no conocía en realidad lo que era trabajar en equipo.

En proyectos típicos de software donde no se toma en consideración ninguna metodología de desarrollo, los tiempos registrados en las diferentes etapas son altos con respecto a proyectos que utilizan TSP para ser desarrollados. Si tomamos como ejemplo la desviación del programa de trabajo del proyecto, ésta se debió a los continuos cambios que se encontraron en las distintas etapas del proyecto. En casos como estos, el equipo de trabajo debe ajustarse para mantener el estado del plan. Como se observa en la Tabla 11, la comparación de la programación en proyectos con TSP y sin TSP es relevante debido a que TSP ayuda a administrar las tareas mediante un rol especifico.

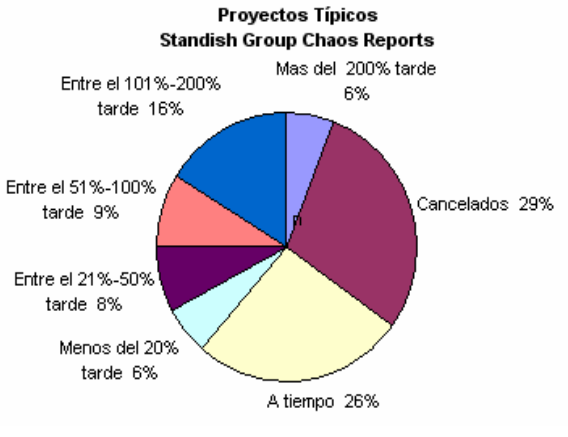
	PROYECTOS CON TSP	PROYECTOS TIPICOS SIN EL USO DE TSP
Desfase promedio en la Programación del trabajo	6%	
Rango aceptable de errores en la programación del trabajo	-20% a 27%	

Tabla No. 11
Comparación de proyectos con TSP y sin TSP (6)

Con respecto al trabajo realizado en este proyecto, la planificación se la dividió por incrementos, los cuales tenían su propia programación de tareas. Inicialmente los resultados fueron catastróficos, obteniéndose un 55.5% de desfase con respecto a lo planificado. Al ejecutar la segunda planificación, el equipo pudo reducir este desfase a 23.7%. Con el siguiente incremento, el grupo de trabajo obtuvo un 4.62% de desfase, y con el último incremento el, equipo obtuvo un 3.47% de desfase. Luego de seguir la metodología TSP, se apreció una mejor organización en el desarrollo de la documentación, se obtuvo una disciplina en el desarrollo del trabajo y los procesos se tornaron más controlados. Se aprendió a estimar y planificar los tiempos de cada administrador, así como desarrollar el proyecto de manera más efectiva, efectuando un buen análisis y diseño de los sistemas.

En el desarrollo del proyecto se presentaron diferentes problemas debido a que el grupo de trabajo no tenía antecedentes de proyectos desarrollados con este tipo de metodología. No se lograba estimar apropiadamente los tiempos pues no se contaba con la experiencia para poder estimar fechas de entrega. Entonces, la utilización de métricas en el proceso de desarrollo de software ha permitido tomar acciones correctivas a tiempo, mejorando sucesivamente los procesos definidos.

Conclusiones

La metodología TSP puesta en práctica en el equipo contribuyó a que el grupo tenga a una mejor comprensión de sus responsabilidades en los procesos. Además, ésta les permite enfocar sus esfuerzos hacia las actividades que son significativas en el desarrollo del proyecto, lo cual les brinda autonomía al reducir el número de interacciones con el instructor. Los resultados preliminares, utilizando procesos rediseñados sugieren que los modelos son una guía poderosa para entrenar a los participantes del proyecto. Los modelos facilitan la colaboración entre los distintos miembros del grupo al determinar explícitamente los tipos de interacción que existen en cada etapa del desarrollo de un sistema de software. El TSP funciona en mejor manera siempre y cuando los miembros del equipo trabajen en un mismo lugar.

La metodología TSP está enfocada a administradores del proyecto, para que exista un mejor seguimiento del trabajo a los desarrolladores, estas funciones deben ser segregadas. La metodología es aplicable para empresas que tengan mayores recursos humanos y logísticos. El registro en las plantillas tanto de PSP como TSP que se utilizaron en el presente trabajo ayudaron a ser más ordenados con el registro en el proceso de desarrollo de software. El modelo incremental es el que más se ajusta a TSP debido a que se puede definir pequeños incrementos y ver las mejoras a medida que se los va desarrollando. El equipo de trabajo es más productivo cuando trabaja en conjunto y no individualmente. Para ello, es indispensable que exista una buena comunicación entre todos los integrantes.

Implicaciones

Se utilizan diversas estrategias en la enseñanza de la ingeniería de software, algunas de ellas se basan sólo en la revisión bibliográfica, sin llevar a la práctica el conocimiento adquirido en un proyecto (7). Otras se basan en el trabajo en equipo, y su objetivo central es que el estudiante se ejercite en el desarrollo de un producto para un cliente real, tome decisiones de acuerdo a las opciones o recursos disponibles y se enfrente con los aspectos de comunicación y coordinación típicos del trabajo en grupo (8). Entonces, un problema recurrente es que el éxito de los proyectos en estos cursos depende de la habilidad y experiencia del instructor para dirigir proyectos. Es probable que distintos instructores logren resultados diferentes utilizando el mismo modelo o que el mismo instructor, con otro grupo de estudiantes, obtenga resultados dispares. Además, en algunos de estos cursos sólo se presta atención a las características de una buena arquitectura e implantación, sin integrar los aspectos de aseguramiento de la calidad y administración. Estos problemas sugieren que muchos de los proyectos de ingeniería del

software sufren de deficiencias en el proceso de desarrollo de software utilizado por el instructor (9).

Basados en la experiencia que se ha adquirido en esta tesis podemos sugerir varios puntos claves para los futuros ingenieros de software:

Mejorar el proceso de desarrollo de software adoptando metodologías que ayude a producir software de calidad y sujetos a los acuerdos con los clientes.

Para dar seguimientos al proceso de desarrollo, es importante definir inicialmente métricas que ayuden a tener indicadores de desempeño al inicio, en el proceso y al fin del desarrollo del software.

Otro punto a considerar es tener a la mano plantillas para asegurar la organización y control, para tener un mayor grado de eficiencia en los procesos de software. Además, es importante considerar que cuando se adopta el uso de plantillas, la información que se registre en ellas debe ser lo necesario para levantar una evaluación de los procesos, es decir, debe tener campos necesarios y no redundantes.

En clases de cursos orientados a la programación, se debería promover el uso de estándares, guías y normas para implementación, tomando muchos énfasis en la calidad del producto desarrollado.

Además, no es necesario que los estudiantes investiguen cuáles son las actividades que corresponden a sus puestos, ni con quién deben interactuar durante el proceso, sino que deben preocuparse por comprender cómo realizar sus tareas y adaptarlas al proyecto de desarrollo de software en el que están trabajando.

Referencias

1. Mónica Villavicencio Jorge Mazón, Jose Alvear, (2005) Aspectos de la Calidad y Dificultades en la Gestión de Proyectos de Software: “Estudio exploratorio”

2. Montesdeoca César (2005) Universidad de las Américas, Puebla. Tesis Profesional, Disponible en Internet http://www.pue.udlap.mx/~tesis/lis/pelaez_r_jj/capitulo3.pdf, ingresado

3. Humphrey, Watts S. Introduction to the Team Software Process, Addison Wesley Longman, Inc

4. Sommerville Ian, Ingeniería de Software. Séptima Edición, Pearson Education

5. Sommerville Ian, Ingeniería de Software. Sexta Edición, Capitulo 4, Pearson Education

6. http://www.sei.cmu.edu/publications/documents/03.reports/03tr014/03tr014ref

7. Tomayko, J. E. (1987). Teaching a project-intensive introduction to software engineering (Reporte técnico CMU/SEI-87-TR-20 ESD-TR-87-171). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

8. Upchurch, R. L. y Sims-Knight J. E. (1998). In support of student process improvement. En Proceedings of the 11th Conference on Software Engineering Education and Training. Atlanta: IEEE Computer Society Press.

9. Collofello, J. S., Kantipudi y M., Kanko, M. A. (1994). Assessing the software process maturity of software engineering courses. En Proceedings of the 25th SIGCSE Technical

Symposium on Computer Science Education (pp. 16-20). Phoenix, AR: ACM Press.

10. Sergio Eduardo Duran Rubio Puntos Puntos por Función.Una métrica estándar para establecer el tamaño del software No puedo controlar, lo que no puedo medir disponible en: http://www.inegi.gob.mx/inegi/contenidos/espanol/prensa/Contenidos/Articulos/tecnologia/puntosxfuncion.pdf

11. Juan Lloréns, Adoración de Miguel, Antonio de Amescua, Manuel Velasco, Problemática de la reutilización, disponible en: http://www.clikear.com/manuales/rom/

12. MsC. Lic. Ailyn Febles Estrada,Medir el proceso de control de configuración, ¿una utopía para la Industria Nacional de Software?, disponible en: www.inf.udec.cl/revista/ediciones/edicion9/febles.pdf