ISSN 1870-4069

Propuesta de un modelo de integración de PSP y Scrum para mejorar la calidad del proceso de desarrollo en una MiPyME

Mauricio Leonardo Urbina Delgadillo, María Antonieta Abud Figueroa, Gustavo Peláez Camarena, Giner Alor Hernández, Alma Ivonne Sánchez García

Instituto Tecnológico de Orizaba, División de Estudios de Posgrado e Investigación, Veracruz, México

mauricio.urbina.isc@gmail.com, mabud@ito-depi.edu.mx, sgpelaez@yahoo.com.mx, galor@itorizaba.edu.mx, alivsaga@hotmail.com

Resumen. El desarrollo de software dentro del ambiente empresarial se caracteriza por ser rápido, cambiante y agresivo en los tiempos de entrega. Las empresas pequeñas necesitan construir un producto de calidad con pocos recursos para incrementar la satisfacción del cliente. La combinación de métodos ágiles y procesos disciplinados es una opción factible que promete mejorar la gestión del desarrollo de software y la calidad del software. Por esta razón, en este trabajo se presenta una propuesta de un modelo de integración entre Scrum y PSP (*Personal Software Process*) para mejorar la calidad del proceso de desarrollo para MiPyMEs donde los equipos de trabajo sean reducidos y se desea integrar la adaptabilidad y predictibilidad a sus prácticas. El resultado de esta investigación es un modelo de mejorar de procesos de desarrollo de software aplicable cualquier proyecto.

Palabras clave: Calidad de software, gestión del proceso de software, mejora de proceso de software, proceso personal de software, PSP, scrum.

PSP and Scrum based Integration Model to Improve Software Quality Process for MiPyME

Abstract. Software development into the enterprise environment is defined by speed, volatile and aggressive delivery times. Small enterprises need to build a quality product with limited resources for increase customer satisfaction. Combination of agile methods and process disciplines is a feasible option to improve the management of software development and software quality. In this paper we propose an integrated process model between Scrum and PSP (*Personal Software Process*) to improve the quality of the development process for MiPyME's, which have small work teams and want to integrate adaptability and predictability into their practices. As a result of this research, a model for improve software development processes for any project is presented.

Keywords: Software quality, software process management, software process improvement, personal software process, PSP, scrum.

1. Introducción

La demanda de la calidad del software es un punto importante para construir una relación de confianza entre la empresa y el cliente. Según [1] el 70% de la fallas son introducidas en las fases de ingeniería de requerimientos, diseño del sistema, arquitectura y diseño de los componentes; el otro 30% corresponde a las fases de codificación, pruebas e integración. Aún cuando el 50.5% de esas fallas sean encontradas en las fases de pruebas e integración, todavía el 21% son encontrados en la fase operativa, es decir, por parte del usuario. Este problema surge de dos puntos de vista; el primero se refiere a la demanda del tipo de aplicaciones y la complejidad que éstas presentan a la hora del desarrollo y, el segundo está relacionado con la competitividad por entregar un mejor producto en poco tiempo, de bajo presupuesto y que cumpla con las características del cliente.

La combinación de métodos ágiles (*SCRUM, XP, Lean Development (LD), Ágile*) y disciplinados (*CMMI-DEV, Six Sigma* y *PSP/TSP*), es una solución cada vez más aceptada por los ingenieros de software, para contrarrestar las deficiencias que traen consigo el utilizar un solo método [2], proporcionar mejoras al desarrollar un producto de calidad, mejorar la gestión del proceso y satisfacer las necesidades del cliente.

Esta investigación está enfocada en la combinación entre Scrum y PSP para mejorar la calidad del desarrollo de software. Por esta razón se propone un modelo de procesos para integrar la agilidad y flexibilidad de Scrum con la disciplina que caracteriza a PSP, con el propósito de mejorar el proceso para desarrollo de software para MiPyMEs. Este documento se estructura de la siguiente manera: en la Sección 2 se mencionan trabajos relacionados con la investigación. En la Sección 3 se presentan la justificación de la selección de Scrum y PSP, además de una pequeña definición de éstos. La Sección 4 presenta la propuesta de integración entre Scrum y PSP. Las conclusiones y las líneas de trabajo futuro se muestran en la Sección 5.

2. Trabajos relacionados

Para conocer una perspectiva clara y sencilla de los trabajos relacionados se presenta en la Tabla 2, un resumen de las diferentes adopciones por parte de empresas de software con modelos disciplinados y/o métodos ágiles.

Tabla 1. Análisis comparativo entre trabajos con diferente adopciones entre métodos ágiles y métodos disciplinados.

Objetivo / Planteamiento	Solución	Resultados
Lina, 2012 [3]		
Realizar una correlación entre *CMMI* y *Scrum*, mostrando diferencias entre ellos e identificando como las PyMEs están adoptando prácticas en sus proyectos para hacer estos enfoques más dóciles.	La integración de las siguientes actividades: *Nivel del Proceso de la Organización*, *Administración de las Actividades del Proyecto*. *Gestión de Riesgos*, *Soporte a la Gestión de Configuración de Procesos*, *Aseguramiento de la*	CMMI y Scrum puede complementarse una a la otra creando una sinergia que beneficia a la organización que los utiliza. Resultando útil para organizaciones que tiene un proceso basado en métodos disciplinarios y planean

Objetivo / Planteamiento	Solución	Resultados
	Calidad y *Construir un Nuevo Ciclo de Vida de Software basado en Scrum*.	mejorar la agilidad de los procesos.
Bougroun, 2014 [4]		
Proyección entre CMMI 3 y tres prácticas ágiles: Scrum, Kanban y XP. Su planteamiento parte de otros autores que realizaron análisis para determinar la compatibilidad entre métodos ágiles y CMMI.	Mapeo extenso y detallado de áreas de CMMI: *Desarrollo de requerimientos*, *Solución Técnica*, *Integración del Producto*, *Verificación*, *Validación*, *Concentra en el proceso de la organización*, *Definición del proceso de la organización*, *Entrenamiento de la Organización*, *Gestión de la integración del proyecto*, *Gestión de riesgos*, *Resolución y Análisis de decisiones*.	Se muestran una cobertura, de CMMI Niv. 3, con Scrum del 44% involucrado en las áreas de administración y procesos de la organización, con XP con el 45% en los detalles e implementación de procesos y con Kanban con el 6% en el aspecto de control y decisiones.
Brown, 2014 [5]		
Las diferentes adaptaciones de prácticas de PSP con *Scrum*, *PSP* y *XP* dificultan e impiden las reutilización en otras metodologías. Generar un modelo utilizando el núcleo SEMAT para eliminar esta limitante.	Se muestra a PSP como una alpha "*way of working*". Los 7 niveles de PSP son representados como estados de un sub-alpha "*PSP Compliance*". Esta representación da a conocer a todo el equipo en qué nivel PSP se encuentran. Igualmente se adapta Scrum utilizando el Backlog, el incremento y el Sprint como alphas dentro de SEMAT.	Esta solución se adapta a Scrum donde su capacidad de adaptación se aprecia, debido a que la representación de las actividades de PSP que tienen que ser realizadas por el equipo.
Romano, 2015 [6]		
Aborda el uso de Scrum dentro de una empresa pequeña. Proporciona actividades de monitoreo para la gestión de proyectos de software, lo que implica el uso de recursos humanos, financieros y temporales durante el desarrollo de un proyecto, aumentado la velocidad de sus entregas exitosas.	Una vez definido el método ágil se llevó a cabo su realización, en una empresa de Brasil, fue necesario mencionar los pasos del ciclo: *Selección de la Infraestructura*, *El Equipo de Entrenamiento*, *Despliegue de Scrum*, *Refinamiento del Despliegue*.	Puntos positivos: mejora en la calidad y tiempo estimado, se fortaleció el trabajo en equipo, los desarrolladores reconocieron las deficiencias por la falta de un método para la toma de decisiones, el uso *Kanban* facilito la concentración y comprensión de los objetivos y tareas, *Planning Poker* es de utilidad para las estimaciones de

Objetivo / Planteamiento	Solución	Resultados
		tiempo y el gráfico *Burndown* muestra el progreso del Sprint y las tareas que se tiene que completar.
Soares, 2015 [7]		
Las iniciativas de transición a métodos ágiles fallan debido a que los empleados creen que el proceso no encaja, el cambio es obligado y sin consultarlo antes. Basándose en este escenario de define una estrategia ágil que pueda ayudar a las empresas a implementar prácticas de administración de proyectos.	En total, esta propuesta definió 84 maneras para implementar, de una forma ágil, la práctica de gestión de proyectos y 38 productos de trabajo como parte de la estrategia para implementación de la administración de proyectos agiles para empresas que buscan CMMI.	Para guiar a las empresas dentro este tipo de escenarios, este trabajo presentó una estratégica que apoya de manera gradual y disciplinada la puesta en marcha de la gestión de proyectos agiles basado en marcos de trabajo y métodos previamente validados y en su creciente uso ara la comunidad de desarrollo de software.
Arauz Ortiz, 2016 [8]		
Describe la experiencia de una empresa mexicana de desarrollo de software que integra métodos ágiles dentro del proceso de desarrollo CMMI-DEV Niv. 5. El objetivo del caso de estudio fue documentar los efectos, cuantitativos y cualitativos, reflejados en el desempeño de sus proyectos.	La motivación fue conocer las técnicas ágiles que integró Praxis (organización en el área de Tecnologías de la Información) con CMMI-DEV para establecer una nueva forma de construir software y mejorar la productividad. Las unidades analizadas fueron: *Técnicas ágiles integradas con CMMI-DEV*, *Métricas de proyectos para determinar beneficios*, *Adaptación del grupo de trabajo*.	Los efectos de la integración tuvieron un impacto sobre los integrantes de los equipos, el cliente y la misma organización. Algunos beneficios obtenidos fueron: *Enfoque en producto y no en documentación, mayor control de actividades, corrección oportuna de defectos, aumento de productividad, conservar la calidad, mejoras en el ambiente de trabajo, satisfacción del cliente*.

3. Modelos de desarrollo de software: Scrum y PSP

3.1. Justificación

Como se puedo observar en el apartado anterior, la mayoría de las empresas utilizan CMMI como modelo de madurez y Scrum como método para la administración ágil del proyecto. El problema que enfrenta las MiPyMEs para implementar CMMI es la carencia de un proceso y la gran cantidad de áreas que define el modelo de madurez; por otro lado, Scrum es perfecto para cualquier tipo de empresas, en especial las MiPyMEs, para agilizar el proceso de desarrollo sin sobrepasar los tiempos y recursos estipulados por los clientes sin olvidar la gestión y la calidad del producto. Scrum ha sido evaluado como un “muy buena” práctica

para proyectos de tamaño pequeño y mediano con un equipo de trabajo relativamente pequeño (3 a 9) [9,10]. Por otro lado PSP, define prácticas sencillas y ligeras, considera una "excelente" práctica para proyectos de todo tipo de tamaño aplicable a nivel personal (1 a 3) [10] que favorecen la institucionalizan de procesos para este tipo de empresas. Scrum y PSP comparten los equipos pequeños, multidisciplinarios y auto-dirigidos para producir software; aquellos que utilizan Scrum usan las buenas prácticas para construir un producto adecuado y PSP ayuda a construir el producto correctamente.

3.2. Scrum

Desarrollador por Ken Schwaber and Jeff Sutherlan basado en el concepto de que el desarrollo de software no es un proceso definido, sino un proceso empírico con transformaciones de entrada y salida complejos que pueden o no repetirse en circunstancias diferentes [9]. Es definido como un marco de trabajo por el cual las personas pueden trabajar con problemas complejos adaptativos, al mismo tiempo se entregan productos de gran valor posible [11]. Scrum implementa un ciclo de vida iterativo e incremental, formado por tres fases *Pre-juego*, *Juego* y *Post-juego* el cual involucra tres roles en particular: *Product Owner* (Dueño del Producto), el *Equipo de Desarrollo* y el *Scrum Master* [11, 12] mostrados en la Figura 1.

Scrum hace énfasis en la administración del proceso del desarrollo dirigido por el *Scrum Master* [2], la base de la planeación comienza con la generación de la lista de requisitos (*Product Backlog*) que contiene las mejoras funcionales y tecnológicas para el proyecto. Existe un ciclo de desarrollo correspondiente a 30 días, llamado *Sprint*, el cual es precedido por las actividades de las fases *Pre-juego* y *Post-juego*. Adicionalmente, el objetivo del *Sprint* es establecido, el cual sirve como un criterio mínimo para el éxito del mismo y mantiene al equipo enfocado en panorama global y no estrictamente en tareas específicas.

Se definen tres reuniones para el *Sprint*, la primera es una junta (30 minutos) que permite al equipo monitorear el estatus y los problemas de comunicación del proyecto, la segunda se realiza para mostrar el avance al *Product Owner* para obtener una observaciones, dudas o cambios sobre el producto, la tercera es una retroalimentación sobre lo bueno y lo malo de proceso utilizado en el anterior ciclo y se propone actividades de mejora.

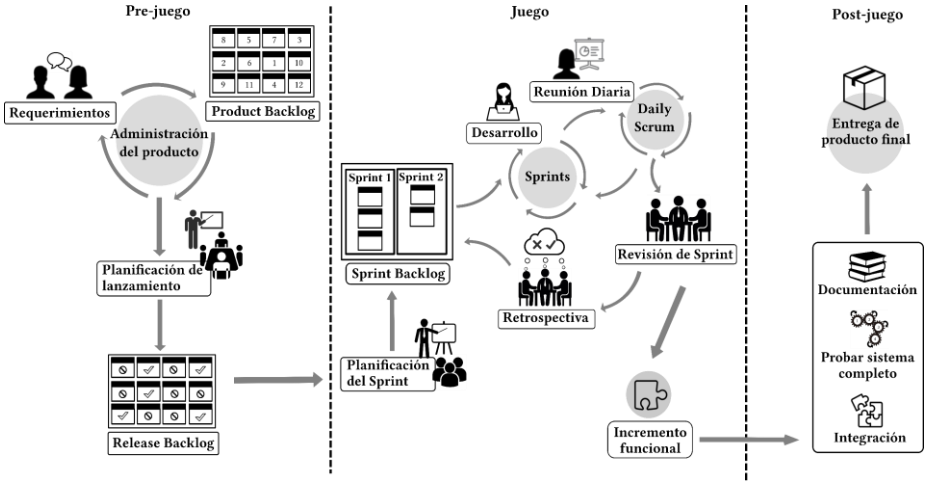

Fig. 1. Ciclo de vida de Scrum.

3.3. *Personal Software Process* (PSP)

Con base a las experiencias y los datos de cursos iniciales, Watts S. Humphrey escribió y publicó el primer libro sobre *Personal Software Process* (PSP) a finales de 1994. PSP es definido con un marco de trabajo para mejorar los procesos de construcción de software que funciona a nivel personal, su estrategia permite mejorar el rendimiento de los programadores utilizando prácticas sólidas para monitorizar y esforzarse en mejorar el desempeño, y la calidad de los productos. PSP se basa en los siguientes principios [13]:

1. Para una mejorar continua del desempeño, los desarrolladores definen y miden correctamente sus procesos.
2. Para producir productos de calidad, los desarrolladores deben sentirse responsables de la calidad de su trabajo.
3. El costo es menor si se encuentran y reparan defectos en fases tempranas que en fases próximas.
4. Es más eficiente prevenir los defectos que buscarlos y repararlos.
5. El camino correcto el siempre el más rápido y barato de trabajar.

PSP define siete niveles de madurez como se muestra en la Figura 2. Los niveles PSP0 y PSP0.1 son críticos para aprender la disciplina del proceso, considerados punto de partida para la secuencia del proceso. Cada nivel de PSP se divide en procesos descritos por medio de "*scripts*" los cuales no solo especifican los pasos a seguir, también los criterios de entrada y salida.

4. PSP y Scrum: integrando dos enfoques en el desarrollo de software para MiPyMEs

4.1. Análisis de características

En la Tabla 1 se muestra un análisis realizado con el propósito de determinar las características de integración para ambos enfoques considerando factores como: enfoque, tipo de conocimiento, prácticas para el desarrollo, estructura/fases, ambiente de trabajo, tipo de proyecto y cultura.

Tabla 2. Análisis entre factores para Scrum y PSP.

Factores	***Scrum***	***PSP***
Enfoque	Agilidad, adaptación a cambios e inspecciones de trabajo. Producción rápida.	Mejorar las habilidades personales del ingeniero de software. Estimación basada en datos históricos. Predictibilidad estadística.
Tipo de conocimiento	Empírico, Tácito.	Teórico basado en registros.
Prácticas en el desarrollo	Gestión del proceso centrado en los requerimientos del cliente. Priorización de los requerimientos y estimación del tiempo. Descomposición de tareas.	Establece un flujo de trabajo para el ingeniero de manera personal. Define guías (*scripts*) para la administración del proceso.

Factores	*Scrum*	*PSP*
Estructura/Fases	Iterativo e incremental. Define tres fases: *Pre-juego*, *Juego* y *Post-juego*.	Iterativo y escalonado por niveles. Define tres principales procesos: *Planificación*, *Desarrollo* y *Postmortem*.
Ambiente de trabajo	Cambiante, inquieto, rápido, enfocado al proyecto.	Estable, pocas modificaciones, enfocado a los procesos.
Tipo de proyecto	Para proyectos cortos, medianos. Equipo de trabajo pequeño, poca necesidad de documentación.	Para cualquier tipo de proyectos, se enfoca en el trabajo del ingeniero. Gran cantidad de datos se registran en formularios.
Cultura	Trabajo y colaboración en equipo. Permite una mejora continua para todos los integrantes del equipo.	Establece la disciplina y el respeto al proceso de trabajo. Establece una mejora continua de las habilidades personales del desarrollador.

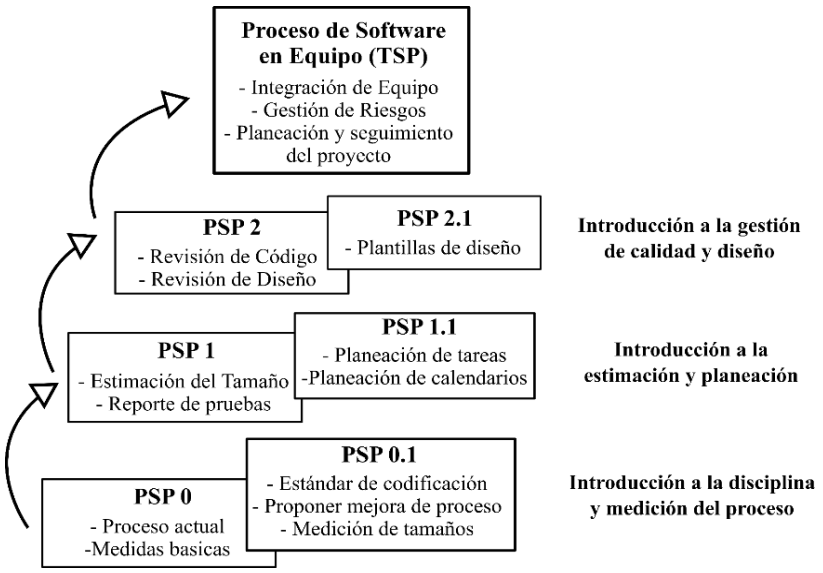

Fig. 2. Niveles de Madurez de PSP (Fuente: PSP: A Self-Improvment Process for Software Engineers, p. 8).

Una desventaja encontrada en Scrum es la carencia de guías para el proceso de desarrollo dirigidas al ingeniero de software en la construcción el incremento en los *Sprints*, PSP ayuda a eliminar esta brecha proporcionando prácticas sencillas para la construcción del incremento siguiendo los requisitos del cliente. Gracias a la recolección de datos, por parte de PSP, se generan registros donde la información se utiliza para predecir un mejor tiempo de trabajo y genera un plan de mejora individual.

La esencia de Scrum es: la agilidad, flexibilidad a cambios, estimación de tiempo de actividades y un proceso iterativo e incremental. Estas características son consideradas parte fundamental del esqueleto de la integración.

4.2. Modelo de integración: Scrum y PSP trabajando juntos

El siguiente modelo es una propuesta para integración de Scrum y PSP enfocada hacia las MiPyME's donde los equipos de desarrollo son pequeños. El modelo de integración entre Scrum y PSP está formado por dos capas. Antes de comenzar el desarrollo de un proyecto, se debe tener en cuenta las siguientes actividades preliminares:

- *Identificar la oportunidad apropiada* para el desarrollo de un sistema de software, que representa la comprensión compartida de las necesidades de los involucrados.
- *Conocer el tipo de proyecto* que se llevará a cabo, desarrollo o mantenimiento de software.
- *Reconocer a todos los posibles grupos diferentes de involucrados* que son, o serán afectados por el desarrollo y operación del sistema de software, junto con las responsabilidades de cada uno.

Ciclo de vida. Esta capa está definida por las tres fases: *Preparación*, *Desarrollo* y Entrega compaginados con las fases de *Pre-game*, *Game* y *Post-game* de Scrum respectivamente. El cambio en los nombres se realizó para una rápida identificación por parte del personal que lo implementan y sea más fácil su flujo de procesos. En la Figura 3 se muestra el ciclo de vida de la *Integración Scrum-PSP*.

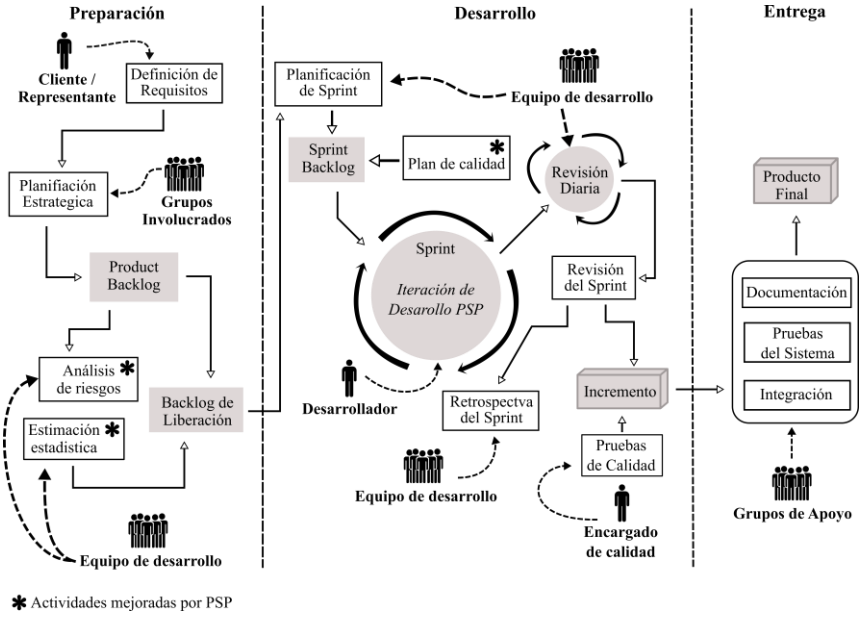

Fig. 3. Ciclo de vida del modelo de integración Scrum-PSP.

La fase de *Preparación* abarca desde la definición de las necesidades del cliente, que posteriormente se convertirán en *Product Backlog*, hasta la creación del *Backlog de liberación*. El proceso de captura de requisitos no exige herramientas y/o formatos para documentar, solo aconseja que los requerimientos sean lo más claros posibles, sin ambigüedades y accesibles por todos los grupos involucrados respetando su perfil de seguridad. La planificación estratégica se realiza respetando la lista de requerimientos y,

conforme los requisitos son priorizados y estimados por el equipo formando el *Backlog de Liberación*. Durante esta misma actividad se realiza el diseño de la "Arquitectura de Alto Nivel" que ayude al equipo como abordar los requerimientos. El "Análisis de Riesgos" será reforzado por las habilidades que PSP desarrolle en los integrantes y las estimaciones estarán apoyadas por el método estadístico PROBE, el cual tiene buena precisión para calcular el costo de las actividades en esfuerzo y tiempo, según los informes del *Software Engineering Institute* (SEI).

La fase de *Desarrollo*: El equipo de desarrollo tiene que definir el número de *Sprints* que se utilizarán para desarrollar el incremento y tiene que crear el *Sprint Backlog* para cada iteración.

Un "Plan de Calidad*"* tiene que ser generado por parte del equipo y respetado hasta el final del desarrollo del incremento; en un principio este plan se realizará de manera abstracta y conforme los ingenieros adopten la disciplina de PSP dicho plan será cada vez más detallado. Además, se integran las prácticas de PSP basadas en las tres principales fases: "*Planeación*, *Desarrollo* y *Postmortem*. El *Scrum Daily* continúa siendo una actividad sin cambios proveniente de Scrum.

La fase de *Entrega*: Establece la integración del nuevo incremento con el incremento generado en la anterior iteración. Se realizan pruebas al sistema completo para verificar su funcionamiento en diferentes entornos. Se realizan las configuraciones necesarias para que el nuevo sistema funcione los equipos de cómputo del cliente. Un punto importante de esta fase es que la documentación del proyecto tiene que estar terminada y verificada por las personas involucradas.

Iteración de desarrollo PSP. Es el proceso en el cual cada ingeniero de software utiliza las actividades clásicas para construir los programas o módulos correspondientes a sus actividades asignadas y está ubicada dentro del *Sprint*. La Figura 4 muestra la estructura del flujo de trabajo para PSP la cual consiste en 6 procesos.

- *Planificación personal:* Se produce un plan detallado para trabajar el desarrollo del programa definido por los requisitos del problema, los formatos para escribir el plan de trabajo no son difíciles pero requieren toda la atención del desarrollador. El plan consiste en la obtención y definición de los requerimientos para el programa escritos en documento claramente y sin ambigüedades.
- *Diseño Detallado:* Se realiza un diseño detallado para las especificaciones del programa definido por los requerimientos, las herramientas utilizadas es responsabilidad del desarrollador.
- *Código:* La transformación del diseño a sentencias de lenguajes de programación.
- *Compilación:* Se traducen las sentencias del lenguaje de programación a código ejecutable. La mayoría de los defectos de sintaxis serán removidos durante esta fase. Esta fase es *"opcional"*, determinada por el entorno de desarrollo y el lenguaje de programación.
- *Pruebas Unitarias:* Cada desarrollador realiza pruebas unitarias al programa o módulo para verificar que cumpla con los requerimientos, no se establece un número límite para las pruebas o herramientas para realizarlas, sin embargo se tienen que registrar el tipo de cada prueba realizada.
- *Postmortem:* Considerada como una retrospectiva personal para resumir y analizar los datos generados por el proceso. Estos datos incluyen valores sobre el tiempo estimado y

el tiempo real utilizado, calidad y productividad. Además, la información proporcionada por los productos personales permite dar bases a las retrospectivas del *Sprint*.

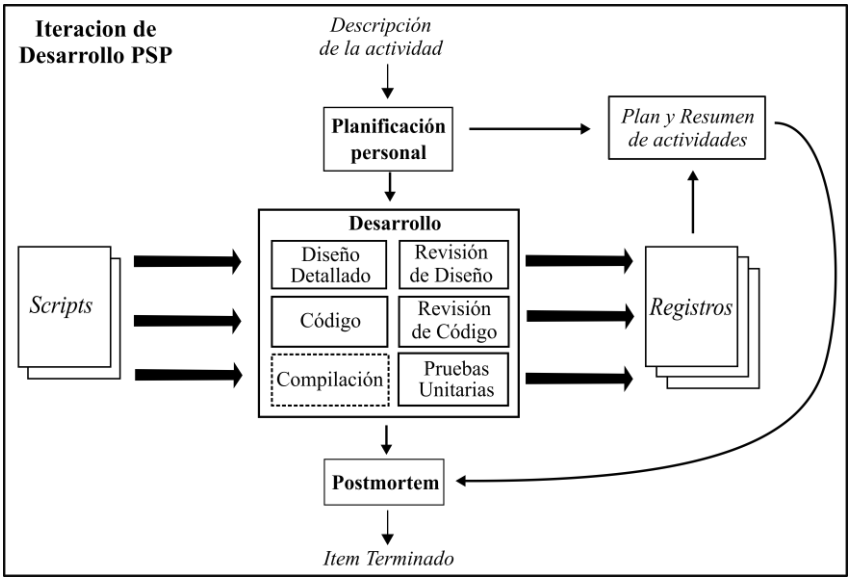

Fig. 4. Iteración de desarrollo PSP.

5. Conclusiones y trabajo a futuro

En esta investigación se ha propuesto un modelo de integración entre Scrum y PSP preservando la esencia de ambos enfoques para mejora el trabajo individual del equipo, introduciendo prácticas que fomentan la disciplina del proceso y la institucionalización de los procesos de una empresa. Se espera que las prácticas de PSP ayuden a Scrum en generar estimaciones de tiempo más exactas, gracias al registro de datos y defectos. Un análisis de riesgos más preciso gracias a la disciplina creada por el proceso de PSP. Reducir el tiempo invertido en la etapa de pruebas gracias a las revisiones de diseño y código.

Como trabajo futuro se tiene la realización de casos de estudios en diferentes ambientes, un equipo con integrantes con experiencia en Scrum y otro integrado por estudiantes para conocer los puntos positivos y negativos de la propuesta. Comprar los resultados de individuos que utilizan PSP con aquellos que usan otros procesos para verificar si existen mejoras en la calidad del producto y calidad del proceso. Establecer una extensión o modificación al modelo propuesto para aquellas empresas o equipos que se encargar exclusivamente del mantenimiento del software. Enriquecer el modelo con la adicción de normas de calidad como: *ISO/IEC*, niveles de *CMMI-DEV*, prácticas de *MoProSoft* o *CompetiSoft*. Enriquecer la dinámica del trabajo en equipo con prácticas de "*Team Software Process*" (TSP).

Agradecimientos. Esta investigación fue apoya por El Consejo Nacional de Ciencia y Tecnología (CONACYT) bajo el Programa Nacional de Posgrados de Calidad (PNPC) y el Instituto Tecnológico de Orizaba (ITO) División de Estudios de Posgrado e Investigación (DEPI).

Referencias

1. Nielsen, P.: Struggles at the Frontier of Software Engineering. In: TSP Symposium, Going beyond agile, D.F., México (2016)
2. Boehm, B., Turner, R.: Balancing agility and discipline. Addison-Wesley, Boston (2004)
3. Lina, Z., Dan, S.: Research on Combining Scrum with CMMI in Small and Medium Organizations. In: International Conference on Computer Science and Electronics Engineering (2012)
4. Bougroun, Z., Zeaaraoui, A., Bouchentouf, T.: The projection of the specific practices of the third level of CMMI model in agile methods: Scrum, XP and Kanban. In: Third IEEE International Colloquium in Information Science and Technology (CIST) (2014)
5. Brown, D. et al.: PSP Implementations for agile methods: a SEMAT-based approach. Software Engineering: Methods, Modeling, and Teaching, 3, pp. 41–45 (2014)
6. Romano, B., Silva, A.: Project Management Using the Scrum Agile Method: A Case Study within a Small Enterprise. In: 12th International Conference on Information Technology - New Generations (2015)
7. Soares, F., de Lemos Meira, S.: An agile strategy for implementing CMMI project management practices in software organizations. In: 10th Iberian Conference on Information Systems and Technologies (CISTI) (2015)
8. Arauz, Ortiz, G., Morales Trujillo, M., Oktaba, H., Ramirez Hernandez, E.: Integrating Agile Methods into a Level 5 CMMI-DEV Organization: a Case Study. IEEE Latin America Transactions, 14, pp. 1440–1446 (2016)
9. Sutherland, J., Schwaber, K.: The Definitive Guide to Scrum: The Rules of the Game. Scrum. Org and ScrumInc (2014)
10. Jones, C.: Software engineering best practices. McGraw-Hill, New York (2010)
11. Palacio, J., Ruata, C.: Scrum Manager Gestión de Proyectos. 4th ed. SafeCreative, pp.57–87 (2010)
12. Schwaber, K.: SCRUM Development Process. Business Object Design and Implementation. pp. 117–134 (1997)
13. Humphrey, W.: PSP: A Self-Improvment Process for Software Engineers. Upper Saddle River, NJ: Addison-Wesley (2005)