Cuspe: propuesta ágil para la construcción de software basada en TSP*[1]

[Cuspe: Agile approach for software development based on TSP]

Giovanni HERNÁNDEZ [2], Álvaro MARTÍNEZ [3]

Recibo: 20.11.2013 – Aprobación: 04.04.2014

Resumen

El objetivo principal de este artículo es aportar a la forma de trabajo de las empresas de la industria de software en Nariño mediante una propuesta metodológica ágil basada en Team Software Process (TSP). Este trabajo se desarrolló bajo el paradigma cuantitativo, con un enfoque empírico-analítico de tipo descriptivo-propositivo. La población objeto de estudio fueron las metodologías para construir software basada en equipos de trabajo, de las que se seleccionó de manera intencional TSP. Como resultado, se estableció que el nivel de agilidad que posee TSP corresponde al 79.25%. Así mismo, se diseñó Cuspe, una propuesta para intervenir los principios del manifiesto ágil que presentan el menor nivel de cumplimiento en TSP. El trabajo permite concluir que la propuesta logra incrementar el nivel de agilidad de la metodología y busca la entrega temprana y frecuente de software con valor, a través del trabajo y compromiso de personas motivadas; y consigue medir el progreso.

* **Modelo para la citación de este artículo:**
HERNÁNDEZ, Giovanni & MARTÍNEZ, Álvaro (2014). Cuspe: propuesta ágil para la construcción de software basada en TSP. En: Ventana Informática No. 30 (ene-jun). Manizales (Colombia): Facultad de Ciencias e Ingeniería, Universidad de Manizales. p. 81-96. ISSN: 0123-9678

1 Artículo de investigación científica y tecnológica proveniente del proyecto *Cuspe: propuesta ágil para la construcción de software basada en TSP*, ejecutado en el periodo *04/-04/2013*, e inscrito en el grupo de investigación *GISMAR* de la *Universidad Mariana.*

2 Ingeniero de Sistemas, MSc. en Docencia Universitaria. Docente asociado tiempo completo, Universidad Mariana (Pasto, Nariño, Colombia). Correo electrónico: gihernandez@umariana.edu.co

3 Ingeniero de Sistemas, MSc. en Docencia Universitaria. Docente asociado tiempo completo, Universidad Mariana (Pasto, Nariño, Colombia). Correo electrónico: amartinez@umariana.edu.co

Palabras Clave: *Construcción de software*, *Manifiesto por el Desarrollo Ágil de Software, Team Software Process*

Abstract

The main goal of this article is to support Nariño's' Software industry's way of work through a methodological proposal based on Team Software Process (TSP). This work was carried out under a quantitative paradigm and a descriptive-propositional-based empirical-analytical approach. The object populations were the methodologies for building software based on work teams, in which TSP was intentionally chosen. As a result, it was set that TSP's level of agility reached 71.25%. Furthermore, it was designed Cuspe as an proposal in order to intervene the principles of the Agile Manifesto which shows the lowest level of accomplishment regarding TSP. This work allows concluding that the proposal increases the level of methodology's agility and leads to an early and frequent valuable software production through the work and compromise of motivated people, and also progress measurement.

Keywords*: Manifest for Agile Software Development, Software Development, Team Software Process.*

Introducción

El interés por elaborar una propuesta ágil para construir software basada en TSP, nace a partir de la necesidad de las empresas de la Industria de software por obtener productos de calidad, y de manera rápida. Según Hernández & Martínez (2012, 16), las empresas del sector productivo en Nariño, están adquiriendo software y servicios de TI a través de *outsourcing*. No obstante, para contratar este tipo de servicios, estas organizaciones están exigiendo a los proveedores certificaciones que brinden garantía de la calidad de los productos y servicios prestados. Por esta razón, las empresas de base tecnológica han tenido que establecer alianzas con empresas nacionales que poseen estos requisitos y mayor trayectoria en el sector para participar en los procesos de contratación y buscar oportunidades fuera de su entorno; esta necesidad de unirse se produce porque las empresas del sector son pequeñas y especializadas, según señalan Hernández & Martínez (2012, 17). El presente artículo propone una alternativa basada en un modelo que les permita a futuro, en primer lugar, optar por estas acreditaciones; en segundo, que dé respuesta a su tamaño y operación; y finalmente que la adopción de la propuesta sea rápida.

Esta investigación planteó una propuesta ágil para la construcción de software basada en TSP que logre responder a la problemática planteada anteriormente. Para el desarrollo de la investigación se encontraron trabajos investigativos orientados a plantear alternativas para la construcción de software de manera ágil como el trazado por Pino et al. (2006, 6), Becerril et al. (2009, 1), Prasedo, Dolado & Aguirregoitia (2010, 65), Quiroga (2007, 1) y Pardo, Hurtado & Collazos (2010, 251). Por otra parte, existen estudios relacionados con el uso y aplicación de TSP en las empresas de la industria de software como el planteado por Koch (2005, 4), Bustos & Gauallasamin (2007, 1), Ascencio et al. (2009, 1), López & André (2006, 31), Rodríguez et al. (2007, 1) y Flórez (2009, 26), experiencias que permitieron establecer una ruta de trabajo a partir de las lecciones aprendidas en la aplicación de TSP en las empresas de la Industria de software. Además, se revisó la resignificación del concepto de construcción de software por autores como Meyer (1999, 20), Humphrey (2000, 4), así como la consideración de la agilidad, como nueva finalidad en el proceso, desde los aportes de Marco et al. (2010, 194) y Beck et al. (2012, 20).

Teniendo en cuenta el camino teórico que fundamenta la construcción de software basada en equipos de trabajo, se planteó como propósito principal aportar a la forma de trabajo de las empresas de la industria de software en Nariño mediante una propuesta metodológica ágil basada en TSP. Para alcanzar este fin, en una primera fase se describió de manera comparativa los elementos de las etapas y roles en TSP con base en los principios del manifiesto ágil; luego, se diseñó una propuesta de trabajo ágil basada en TSP que abarca instrumentos y herramientas, y finalmente, se determinó el nivel de agilidad de la propuesta en la construcción de un producto software piloto.

La propuesta de trabajo ágil basada en TSP, se convirtió en una alternativa de trabajo para construir software como un proceso perfectible y ágil, cuya utilidad radica en que las empresas de la Industria de Software podrán conocerla, apropiarla y utilizarla como un modelo de trabajo que a futuro logre evolucionar.

Este documento comienza con la descripción del fundamento teórico utilizado para soportar el trabajo investigativo; luego se presenta la metodología, aquí se explica la forma como se desarrolló la investigación. Posteriormente, se muestran los resultados obtenidos y se hace una discusión acerca de algunas consideraciones y reflexiones frente a los hallazgos y finalmente se presentan las conclusiones.

1. Fundamento teórico

El la figura 1 se muestra una jerarquía de teorías que soportan el presente trabajo.

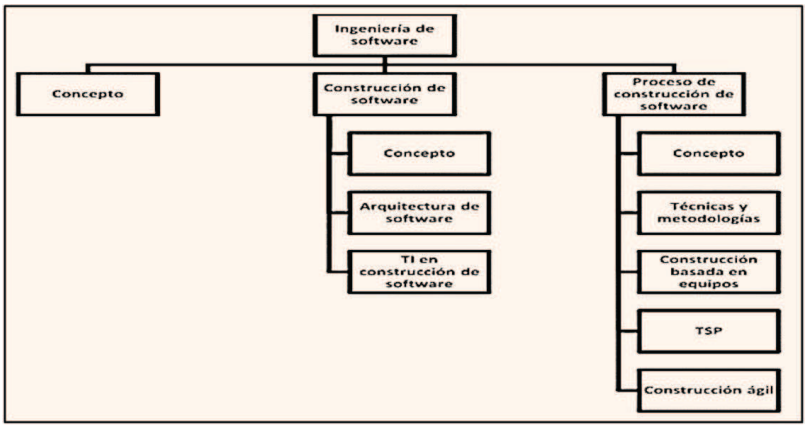

Figura 1. Teorías sobre Ingeniería de Software
(Construida a partir de diversas fuentes)

1.1 Ingeniería de software

Para conceptualizar la ingeniería de software Sommerville (2011, 3) parte de un elemento importante: el software, que además de ser el programa, es el conjunto de documentos asociados y la configuración de datos que se necesitan para hacer que este programa opere de manera correcta. Así mismo para que el software sea de calidad debe incluir atributos como mantenibilidad, confiabilidad, eficiencia y usabilidad que permitan la modificación en caso de que se presente algún cambio; igualmente la seguridad, el buen manejo de los recursos del sistema y la fácil interacción con el usuario deben ser características del producto fabricado. Para definir ingeniería de software, también se deben considerar los procesos de software que involucran modelos como el de roles del equipo de trabajo, cuyo principal objetivo es la asignación de diferentes tareas y responsabilidades a personas encargadas de la creación del software.

Por su parte, O'Regan, (2011, 16) define al software como programas y como estructuración de datos. Así mismo hace referencia a un ciclo de vida que se divide en etapas y que se deben tener en cuenta para la construcción: análisis previo, análisis de requisitos, diseño, programación, pruebas y mantenimiento. Según Casallas y Villalobos (2005, 1), para especificar qué es ingeniería de software se debe tener en cuenta cuatro ejes principales que son su columna vertebral: procesos de software y el aseguramiento de la calidad, arquitectura de software

y los elementos estructuradores, metodologías y técnicas y las tecnologías de información.

1.2 Construcción de software

1.2.1 Concepto. Según Meyer (1999, 20), para definir la construcción de software se debe tener en cuenta que el principal aspecto es su calidad que contiene factores que definen exactamente el concepto: la corrección, la robustez, la extensibilidad, la reutilización, la compatibilidad, la eficiencia, la portabilidad, la facilidad, la funcionalidad y la oportunidad. En síntesis, la construcción de software es el proceso mediante el cual se realiza un producto software bajo ciertos estándares de calidad.

1.2.2 Arquitectura de software. Fernández (2006, 40), para definir Arquitectura de software parte de tres componentes: elementos, forma y razón. Por su parte, Bass, Clements y Kazman (2003, 43) la definen partiendo de la importancia que tiene, debido a que permite la comunicación entre las personas involucradas en la construcción del producto y así obtener una comprensión adecuada, además permite tomar las decisiones más convenientes en el diseño, la implementación y el mantenimiento del ciclo de vida del software después de un análisis y un estudio. Esta disciplina sirve para la comprensión de sistemas grandes y complejos, la reutilización de diseños ya creados para la construcción de otro software y proporciona un mejor entendimiento entre las personas que desarrollan el proyecto y su perfeccionamiento.

1.2.3 Tecnologías de la información en la construcción de software. Según Casallas y Villalobos (2005, 3) las tecnologías de información en la construcción de software son importantes en la actualidad ya que su progreso conlleva a la exigencia de cambio constante en el software. Hoy en día han ido tomando relevancia los requerimientos no funcionales que permiten tomar decisiones sobre qué tecnología se debe utilizar desde las etapas iniciales para la construcción de las aplicaciones, obligando ser ágil al ingeniero de software en sus capacidades para elegir un modelo de tecnología adecuado teniendo en cuenta que la actualización de sus conocimientos debe ir de la mano con la innovación tecnológica.

1.3 Proceso de construcción de software

1.3.1 Concepto. Para saber qué es proceso de software se debe tener en cuenta que proceso como tal, lo componen todos los pasos adecuados para alcanzar una meta u objetivo establecido, estos pasos abarcan desde el desarrollo del software hasta su mantenimiento. Sin embargo como los procesos de software son tan complejos se necesitan modelos con los que se establecen diferentes tipos de proyectos; existen algunos que crean el software desde el principio partiendo desde cero pero también existen otros que buscan agregarle alguna función nueva

a uno ya creado anteriormente, igualmente se pueden hacer proyectos de reingeniería de un sistema ya desarrollado que proponga nuevas tecnologías, o se puede crear proyectos que buscan generar uno o varios componentes de software reutilizable.

Para Sommerville (2011, 27), un proceso de software consiste en establecer actividades para crear un producto software, dichas acciones dependen de personas que toman decisiones y juicios. Además para que un proceso de software sea adecuado se necesita saber cuál es exactamente la funcionalidad que se quiere, conocer el diseño, la implementación, necesidades de cambio y el modelo que lo representa, para así asegurar que el software hace lo que el cliente realmente necesita y mirar desde varias perspectivas el desarrollo del producto.

1.3.2 Técnicas y metodologías en la construcción de software. Para definir técnicas y metodologías Casallas y Villalobos (2005, 3) muestran en su artículo que dentro de la metodología para el proceso de software se enuncia la experiencia de otros en la realización de una actividad donde se necesita tiempo de utilización, evaluación, mejoramiento. Las técnicas y metodologías en el proceso de desarrollo de software se aplican como ayuda en los procedimientos de análisis, diseño, programación, elaboración de pruebas, administración de riesgos, además de puntualizar funciones que colaboran en el proceso, describiendo el modo de efectuar el trabajo, de hacerle un seguimiento al avance, la comprobación de la calidad del software y la validación del producto esperado, además las técnicas y metodologías se utilizan porque puntualizan los pasos concretos para el proceso de la construcción del producto.

1.3.3 Construcción de software basada en equipos de trabajo. Meyer (1999, 20), considera este proceso como parte de la definición de construcción de software, y lo define como un procedimiento que crea un producto bajo unos estándares de calidad, y que mediante el trabajo en equipo es posible distribuir las actividades asignadas a cada integrante según sus habilidades a lo largo del desarrollo del proyecto, y así agilizar la construcción en las industrias.

1.3.4 Team Software Process, TSP. Según Humphrey (2000, 19) se parte del concepto de equipo, entendiendo un equipo como un conjunto de personas que se unen para trabajar por una meta en común, aquí a cada integrante se le asigna funciones determinadas o roles que deben llevar a cabo para cumplir dicha meta. Estos equipos pueden ser de cualquier tamaño a partir de dos personas. Sin embargo, en situaciones prácticas los equipos son más eficaces cuando en él se desarrollan relaciones estrechas entre todos los miembros. Los roles asignados son: líder, planeación, calidad, desarrollo y soporte. De acuerdo con lo expuesto, TSP se puede definir como una metodología para

la construcción de software basada en PSP, un proceso en el que se establecen métodos para realizar una planificación y perfeccionamiento de las técnicas de desarrollo al tiempo que permite gestionar el trabajo individual del ingeniero de sistemas. Además, TSP permite conocer las características tanto de roles como de las etapas que se presentan en la construcción de software.

1.3.5 Construcción de software ágil. La construcción de software es ágil y no está ligada a los métodos tradicionales, de acuerdo con Marco et al. (2010, 194), cuando es más importante la interacción entre los miembros que la utilización de técnicas de desarrollo; la construcción de un software útil y de calidad que una documentación adecuada de un sistema; la contribución con el cliente que la realización del contrato; la capacidad de cambio según se presente cualquier situación que el seguimiento de la planificación.

Beck et al. (2012, 1), consideran que para la construcción de software ágil se debe tener en cuenta el manifiesto ágil compuesto por 12 principios: 1. Satisfacer al cliente mediante la entrega temprana y continua de software con valor. 2. Aceptar que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente. 3. Entregar software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible. 4. Los responsables de negocio y los desarrolladores deben trabajar juntos de forma cotidiana durante todo el proyecto. 5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo. 6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara. 7. El software funcionando es la medida principal de progreso. 8. Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida. 9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad. 10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial. 11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados, y 12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para luego perfeccionar su comportamiento.

Según Canós, Letelier y Penadés (2003, 1), algunos métodos para construcción de software ágil son: *eXtreme Programming*, SCRUM, *Crystal Methodologies*, *Dynamic Systems Development Method*, *Adaptive Software Development*, *Feature –Driven Development*, *Lean*

Development, entre otros, en este trabajo no se profundiza al respecto, únicamente se utilizan los principios del manifiesto ágil.

2. Metodología

La investigación que soporta este artículo se realizó bajo el paradigma cuantitativo, con un enfoque empírico-analítico de tipo descriptivo-propositivo. La población objeto de estudio se construyó con las metodologías para la construcción de software basadas en equipos de trabajo, de las que se seleccionó como muestra, de manera intencional, TSP. Las técnicas para la recolección de información fueron la revisión documental y la encuesta. Para el análisis de la información, se utilizó el análisis documental y la estadística descriptiva. Las variables analizadas fueron: etapa, rol y agilidad.

El trabajo se desarrolló en tres etapas: en la primera, se estableció los elementos de las etapas y roles de TSP con base en los principios del manifiesto ágil mediante una revisión y análisis documental. La segunda, se encargó de formular una propuesta de trabajo ágil basada en TSP que abarque instrumentos y herramientas, para ello se realizó un análisis y revisión documental, y un análisis descriptivo de las encuestas. Finalmente, se identificó el nivel de agilidad de la propuesta en la construcción de un producto software piloto, este proceso se llevó a cabo mediante un análisis descriptivo de encuestas.

3. Resultados y discusión

3.1 Descripción de resultados

Luego de realizar una contrastación de las etapas y roles de TSP con los 12 principios del manifiesto ágil se logró determinar que el proceso TSP consta de ocho etapas a saber: a.) Lanzamiento, b.) Estrategia, c.) Planeación, d.) Análisis, e.) Diseño, f.) Codificación, g.) Pruebas y h.) Postmortem. Para la evaluación del nivel de cumplimiento en las etapas TSP de los principios del manifiesto ágil, se adoptó como criterio de evaluación el cumplimiento de los principios en un 50% en las etapas; en el trabajo hecho se encontró que se cumplen por lo menos en cuatro de ellas.

Teniendo como referente este criterio, se realizó un análisis de frecuencias y se calculó la media y se obtuvo que los principios del manifiesto ágil, se cumplen para las etapas del proceso TSP en un 63.5%. Así mismo, se logró establecer que los principios del manifiesto ágil que se cumplen en un nivel superior al 50% en las etapas de TSP son: 2, 4, 6 y 8 a 12. Para elaborar la propuesta denominada Cuspe, se tuvo en cuenta como elementos de

intervención aquellos principios que no se cumplen en las etapas, que para el caso fueron: a.) El principio 1 que se enfoca en dar prioridad y satisfacer al cliente mediante la entrega temprana y continua de software con valor, b.) El principio 3, que consiste en entregar software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible, c.) El principio 5, que hace referencia a que los proyectos se desarrollen en torno a individuos motivados y d.) El principio 7, que se orienta en la entrega del software funcionando.

Por otra parte, TSP plantea, para el ejercicio de los integrantes del equipo, un conjunto de cinco roles. Al contrastar los cinco roles del proceso TSP con los doce principios del manifiesto ágil, teniendo en cuenta como criterio el 50% en el nivel de cumplimiento de los principios en los roles TSP, se identificó que los primeros se cumplen completamente en cuanto a las habilidades definidas para los roles. Al calcular la media, se evidenció que los principios se cumplen en un 95% para los roles. No obstante, existe un único principio que no logra alcanzar el porcentaje planteado por el criterio y corresponde al número cinco que menciona el desarrollo de proyectos en torno a individuos motivados.

Apoyados en los resultados anteriores, se logró concluir que TSP en relación con los principios del manifiesto por el desarrollo ágil de software, tiene un nivel de agilidad del 79.25% y que los principios del manifiesto ágil que menor nivel de cumplimiento presentan, se sintetizan en el mapa conceptual que se muestra en la figura 2. Aquí nace la propuesta Cuspe cuyo objetivo es establecer una forma de trabajo que permita alcanzar un nivel mayor de agilidad de TSP en cuanto al cumplimiento de los principios del manifiesto ágil, a través de la construcción de software con valor, mediante equipos motivados con entregas tempranas y frecuentes de software funcional y midiendo el progreso del avance.

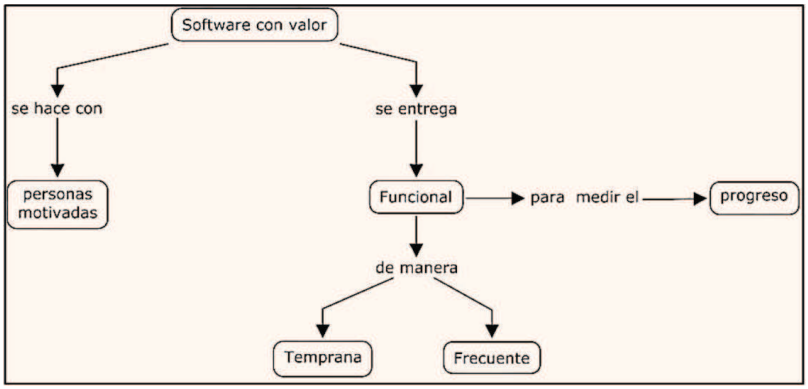

Figura 2. Principios del manifiesto ágil de menor cumplimiento en TSP

La propuesta Cuspe se centra en tres elementos, como se puede observar en la figura 3. El primer elemento corresponde a obtener software con valor a través de entregas frecuentes y tempranas. Para alcanzar este fin, se propone utilizar las Tecnologías de la Información (TI). En este sentido, Kroll y Kruchten (2007, 3) plantean que el Proceso Unificado de Rational (RUP) es un enfoque disciplinado para la asignación y la gestión de proyectos de este tipo.

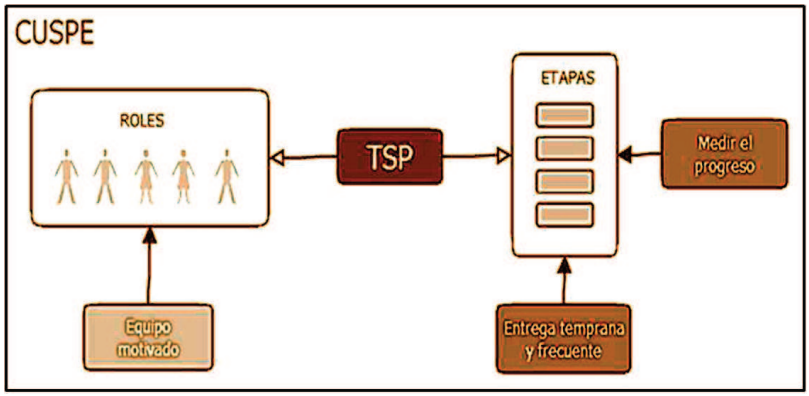

Figura 3. Elementos de Cuspe

Mediante la aplicación de este proceso, los equipos fabrican software de alta calidad y se hace en un tiempo y presupuesto definidos. RUP logra este fin, ya que es una metodología que utiliza UML y sirve para el análisis, diseño, implementación, pruebas y documentación en la construcción de software orientada a objetos. Además, posee un enfoque iterativo, centrado en la arquitectura y los casos de uso. El enfoque iterativo, es un punto de encuentro con Humphrey (2000, 5), quien plantea que construir software a través de iteraciones o ciclos es la respuesta a problemas como: el no conocer en detalle los requerimientos funcionales al inicio de un proyecto, la variabilidad de los requerimientos durante la ejecución de un proyecto y la comunicación entre quien necesita, utiliza y hace el software.

En este mismo sentido, Casallas (2007, 10) manifiesta que el desarrollo a través de iteraciones o incrementos, permite identificar errores de manera temprana, reduciendo costos en el proyecto, de igual manera, se logra incrementar la confiabilidad del cliente ya que el producto software, se empieza a utilizar a partir de la finalización del tiempo definido para el ciclo, por lo tanto, el desarrollo a través de iteraciones o ciclos, obtiene un producto funcional con valor, lineamiento en el que concuerdan RUP y los principios uno y tres del manifiesto ágil. Esto significa que RUP como metodología está

alineada con el cumplimiento de estos principios, en el sentido en que se debe obtener por cada iteración o ciclo un producto software con valor en el menor tiempo.

Por otra parte, RUP propone modelar el software de manera visual y utilizando UML. Este es un factor que al interactuar con el desarrollo iterativo hace un aporte en la reducción del tiempo del ciclo; en otras palabras, si se logra elaborar el diseño del software y esta etapa se la soporta a través de herramientas que la integren con la codificación, se hace un aporte a la reducción de defectos incorporados en el diseño y codificación; y reduce el tiempo del ciclo, por ejemplo, al utilizar un IDE (*Integrated Development Environment*) y poderle incorporar el diseño visual del software utilizando UML, permite generar el código fuente base y agilizar el proceso en las etapas de diseño y codificación.

El segundo elemento que plantea la propuesta Cuspe, como se observa en la figura 2, corresponde a medir el progreso al entregar software funcionando que se hace a través del uso de métricas, entendidas como una medida que permite cuantificar atributos como el tamaño del producto, el estado del proceso, el desempeño de los miembros del equipo, la calidad y el esfuerzo en el desarrollo.

La propuesta Cuspe incorpora el concepto de métrica que, según Pressman, (2010, 527), se trata de la medición de software a través de datos numéricos que sirven para hacer seguimiento y mejorar el proceso de construcción y así medir calidad y proceso de desarrollo, además se incluye en Cuspe, para medir la calidad del progreso, lo que Sommerville (2011, 668) llama métricas de control que logran medir la evolución de la construcción de software cuyos indicadores son: a.) El tiempo planeado, b.) El esfuerzo de los miembros del equipo, c.) Los recursos utilizados y d.) Los cambios que se realicen en el transcurso del desarrollo del producto software. Este elemento se materializó en un artefacto de especificación de métricas para el que se construyó una herramienta software que las administra en cada ciclo de desarrollo de un proyecto de construcción de aplicaciones.

Finalmente, el tercer elemento de la propuesta Cuspe, se encuentra en relación con la entrega de software con valor mediante personas motivadas. En este sentido, Cuspe plantea que la motivación se realice en dos momentos. En el primero, se contextualiza los intereses de los miembros del equipo. A partir de los resultados del primer momento, en una segunda etapa, se plantea estrategias motivacionales teniendo en cuenta los siguientes elementos: la tarea, el ambiente, el equipo de trabajo y el empoderamiento, que se fundamentan en las diferentes teorías expuestas por Gross (2012, 1).

Para Locke, citado por Gross (2012, 2), la tarea consiste en tratar de encontrar sentido y significado a las actividades que se realizan en el equipo. Según Herzberg, citado por Gross (2012, 3), el ambiente se enfoca en los recursos; es decir, los elementos que rodean al equipo deben ser adecuados para desarrollar las actividades. Estos recursos son: el espacio, la infraestructura tecnológica, las formas de comunicación e iluminación. Para Alderfer citado por Gross (2012, 4), dentro de los tres tipos de motivaciones la que se enfoca en el equipo de trabajo es la de relación, que está orientada en las acciones interpersonales, el manejo de conflictos, el mantenimiento de roles definidos y la inexistencia de rivalidad, mantener estas características controladas, permitirán al equipo de desarrollo permanecer unido. Finalmente, McClelland, citado por Gross (2012, 5), plantea que el empoderamiento, se proyecta en dos tipos de motivaciones: la del logro y la del poder, que les permitirá a cada integrante del equipo sentirse impulsados para sobresalir con las metas que le fueron establecidas sin que la competencia sea el fin.

Para la contextualización de los intereses de los roles del equipo, Cuspe se apoya en un instrumento creado por Toro (1992, 1) denominado, cuestionario para la motivación del trabajo – CMT. Este instrumento fue diseñado para identificar y apreciar objetivamente quince factores de motivación. La primera parte del cuestionario está elaborada para proporcionar un perfil de condiciones motivacionales internas a través de cinco factores que son: logro, poder, afiliación, autorrealización y reconocimiento. La segunda parte está representada por cinco factores que son: dedicación a la tarea, aceptación de la autoridad, aceptación de normas y valores, requisición y expectación. La tercera y última parte está representada por los factores: supervisión, grupo de trabajo, contenido de trabajo, salario y promoción.

3.2 Discusión de resultados

Al hacer una contrastación de TSP en relación con los principios del manifiesto por el desarrollo ágil de software, se logró establecer que TSP tiene un nivel de agilidad del 79.25%. En este sentido, TSP como una metodología que plantea un proceso claramente definido y repetible, no se encuentra en sentido contrario a la búsqueda de agilidad en la construcción de software; es decir, un camino para hacer software de manera ágil es la disciplina en el planear, organizar, hacer, verificar y actuar que propone la metodología. Además, hacer uso de TSP involucra elementos que aportan al aseguramiento de la calidad en el proceso como valor agregado. De esta manera, los investigadores creemos en la posibilidad de encontrar caminos que tengan como fundamento la disciplina que plantea el desarrollo de un proceso claramente definido, para lograr agilidad en la construcción de productos software.

No obstante, se debe considerar la existencia de un balance entre las actividades que plantean las etapas y roles de TSP de forma teórica y lo que realizan los equipos de trabajo de forma práctica; en otros términos, apropiar los lineamientos de la metodología de acuerdo con las necesidades y el contexto del equipo posibilitará efectividad en el trabajo, de tal manera que se logre que las actividades realizadas, siempre agreguen valor al proceso y al producto.

Cuspe es una forma de trabajo que se plantea de manera teórica, con el fin de alcanzar un nivel mayor de agilidad de TSP en cuanto al cumplimiento de los principios del manifiesto ágil. Esta propuesta busca que la construcción de software sea con valor, mediante equipos motivados con entregas tempranas y frecuentes de software funcional y midiendo el progreso del avance. Para lograr el cumplimiento de estos principios, se abordó el problema desde los enfoques disciplinares de la Ingeniería de Software y la Sicología Organizacional. En este sentido, se puede afirmar que las complejidades inherentes a la construcción de software, se deben abordar de manera multidisciplinar; ya que existen factores que se encuentran en relación con el pensamiento y comportamiento humano, temáticas revisadas a profundidad desde la sicología organizacional y otras disciplinas. El software es hecho por individuos y por lo tanto, se debe tener en cuenta en su elaboración, las complejidades inherentes a la interacción con otros seres humanos, el ambiente que rodea a las personas, la capacidad de automotivarse y liderar actividades. Por estas razones, los investigadores consideramos importante diseñar propuestas que permitan centrarse no únicamente en el proceso y el producto, sino que además, se incluyan actividades que tengan como eje central a las personas.

La propuesta Cuspe, es un trabajo teórico que tuvo un primer acercamiento a su validación en la construcción de un producto software de tamaño mediano, con un solo equipo de trabajo. Por esta razón, se plantea la validación de la propuesta de manera rigurosa con el objetivo de evaluarla y mejorarla.

4. Conclusiones

La propuesta Cuspe basada en TSP tuvo como finalidad incrementar el nivel de agilidad de la metodología y busca la entrega temprana y frecuente de software con valor, a través del trabajo y compromiso de personas motivadas y midiendo el progreso.

La realización de un análisis y posterior uso de herramientas computacionales que brinden soporte al desarrollo de las etapas y desempeño

de los roles en TSP, permitió efectuar procesos automáticos de gestión de datos que permitieron agilidad en la entrega temprana y frecuente de software con valor.

Identificar y gestionar métricas para medir el progreso del proyecto, permite establecer cuáles son las fallas que se presentan en un ciclo y a partir de los resultados de las mediciones, se puede plantear actividades de mejora para las siguientes iteraciones.

Para desarrollar software con valor es necesario trabajar con personas motivadas, ya que se fortalece el desempeño en un proyecto. Una forma de motivar a los integrantes del equipo es definir y aplicar actividades que permitan el empoderamiento del rol, es decir, que cada miembro del equipo logre sentirse comprometido e identificado con el trabajo que realiza.

Los riesgos son un factor determinante para el éxito de un proyecto de construcción de software, independiente de la metodología a utilizar en el desarrollo del proyecto. Por lo tanto, gestionar los riesgos en un proyecto es un punto clave que minimiza los retrasos en la entrega, aspecto que desmotiva a los integrantes del equipo.

Integrar los elementos de la propuesta Cuspe con TSP fortalece la metodología propuesta, ya que se logra incrementar la agilidad a un nivel alto (86.37%).

Las etapas que cumplen con el mayor número de principios del manifiesto ágil son el lanzamiento, la estrategia, la planeación y el *postmortem*. En contraste, las etapas con menor nivel de cumplimiento de los principios del manifiesto ágil fueron codificación y pruebas.

5. Agradecimientos

Se agradece la colaboración de profesores y directivos del programa de Ingeniería de Sistemas de la Universidad Mariana, quienes aportaron sus experiencias, conocimiento y tiempo. De igual manera, a los ingenieros Jeniffer Cabrera, Isabel Chilanguad, Andrea Portilla y Santiago Guerrero, quienes con su dedicación, sacrificio y entrega hicieron posible la culminación exitosa de este trabajo.

Referencias bibliográficas

ASCENCIO, José; ECHEVERRÍA, Carlos; ECHEVERRÍA, Cynthia & VILLAVICENCIO, Mónica (2009). Implementación de un sistema integrado utilizando Team Software Process. Trabajo de grado (Ingeniero en Computación). Quito (Ecuador): Escuela Superior Politécnica del Litoral.

BASS, Len; CLEMENTS, Paul; & KAZMAN, Rick (2003). Software Architecture in Practice. 2 ed. Boston (EUA): Addison Wesley. 560 p. ISBN: 0-321-15495-9.

BECERRIL, Ariel; DELGADILLO, Laura; MELO, Fernando; NAVARRO, María & ROLDÁN, Juan (2009). KIXA VII, eficiencia en la creación de software para pymes. Trabajo de grado (Licenciado en Ciencias de la Informática). México D.F. (México): Instituto Politécnico Nacional.

BECK, Kent; BEEDLE, Mike; BENNEKUM, Arie; COCKBURN, Alistair; CUNNINGHAM, Ward & FOWLER, Martin (2001). Manifiesto por el Desarrollo Ágil de Software [on line]. Utah (EUA). <http://agilemanifesto.org/iso/es/manifesto.html> [consulta: 11/11/2012].

BUSTOS, Geovanna & GUALLASAMIN, Cristhian (2007). Uso del TSP (Team software Process) en el desarrollo de software. Trabajo de grado (Ingeniero de Sistemas Informáticos y de Computación). Quito (Ecuador): Escuela Politécnica Nacional. 184 p.

CANÓS, José; LETELIER, Patricio & PENADÉS, María Carmen (s.f.). Metodologías Ágiles en el Desarrollo de Software [en línea]. En: Taller "Metodologías ágiles en el desarrollo de Software", en el marco de las VIII Jornadas de Ingeniería del Software y Bases de Datos, JISBD 2003. (12-14/11/20013). Alicante (España): Grupo ISSI, Universidad Politécnica de Valencia.

CASALLAS, Rubby (2007). ¿Aún en crisis? Algunos mitos y desafíos en la Ingeniería de Software [en línea]. En: Sistemas. No. 102 (oct-dic). Bogotá (Colombia): ACIS p. 8-17 ISSN: 0120-5919 <http://www.acis.org.co/fileadmin/Revista_102/columnista.pdf> [consulta: 12/11/2012].

CASALLAS, Rubby & VILLALOBOS, Jorge (2005). El actual Ingeniero de Software [en línea]. En: Sistemas. No. 93 (jul-sep). Bogotá (Colombia): ACIS ISSN: 0120-5919 <http://www.acis.org.co/index.php?id=547> [consulta: 12/11/2012].

FERNÁNDEZ, Luis (2006). Arquitectura de Software. En: Software Guru, Vol. 2, No 3 (may-jun). México DF (México). p. 40-45. ISSN: 1870-0888.

FLÓREZ, Héctor (2009). Procesos de Ingeniería de Software. En: Vínculos. 6, 1 (ene-jun). Bogotá (Colombia): Universidad Distrital Francisco José de Caldas p. 26-39. ISSN: 1794- 211X.

GROSS, Manuel (2012). Pensamiento Imaginactivo [en línea]. Santiago de Chile (Chile). <http://manuelgross.bligoo.com/las-8-teorias-mas-importantes-sobre-la-motivacion-actualizado> [consulta: 12/10/2012].

HERNÁNDEZ, Giovanni & MARTÍNEZ, Álvaro (2012). Evaluación I Foro Regional del Sector Software y Servicios de TI en Nariño. En: Informativo CIP. No. 58 (ene-abr). Pasto (Colombia): Universidad Mariana. p. 16-17. ISSN: 2256-1234.

HUMPHREY, Watts (2000). Introduction to the Team Software Process. Pittsburgh (EUA): Adisson Wesley Longman Inc. 463 p. ISBN: 978-0201477191.

KOCH, Alan (2005). TSP Can be the building blocks for CMMi. En: Cross Talk, The Journal of Defense Software Engineering, Vol. 18, No. 3 (mar). Oklahoma (EUA): Chelene Fortier-Lozancich. p 4-7. ISSN 2160-1577.

KROLL, Per & KRUCHTEN, Philippe (2007). The Rational Unified Process Made Easy, A practitioner›s guide to the rup. Boston (EUA): Person Education. 464 p. ISBN: 978-0321166098.

LÓPEZ, Yucely & ANDRÉ, Margarita (2006). Roles en el proceso de desarrollo de software para las empresas Cubanas. En: Ingeniería Industrial, Vol. 27, No. 1 (mar). La Habana (Cuba): Instituto Superior Politécnico José Antonio Echeverría. p. 31-35. ISSN: 1815-5936.

MARCO, María; MARCO, Josep; BLÁZQUEZ, Josep & SEGRET, Ramón (2010). Escaneando la Informática. Barcelona (España): Editorial UOC. 260 p. ISBN: 978-8497881104.

MEYER, Bertrand (1999). Construcción de Software Orientado a Objetos. 2 ed. Madrid (España): Prentice Hall. 1198 p. ISBN: 978-8483220405.

O›REGAN, Gerard (2011). Introduction to Software Process Improvement. London (UK): Springer. 266 p. ISBN: 978-0857291714.

PARDO, César; HURTADO, Julio & COLLAZOS, César (2010). Mejora de procesos de software ágil con agile SPI process. En: Dyna. Vol. 164 (dic). Bogotá (Colombia): Universidad Nacional de Colombia. p. 251-263. ISSN: 0012-7353.

PINO, Francisco; GARCÍA, Félix; PIATTINI, Mario & OKTABA, Hanna (2006). Revisión sistemática de mejora de procesos de software en micro, pequeñas y medianas empresas. En: Revista española de innovación, calidad e ingeniería del software. Vol. 2, No. 1 (abr). Madrid (España): Asociación de Técnicos de Informática. p. 6-23. ISSN: 1885-4486.

PRASEDO, Concepción; DOLADO, J. Javier & AGUIRREGOITIA, Amaia (2010). Estudio de métricas para el control de proyectos software. En: XV Jornadas de Ingeniería del Software y Bases de Datos, JISBD 2010 (07-10/09/2010), Valencia (España): Sociedad de Ingeniería de Software y Tecnologías de Desarrollo de Software. Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, Vol. 4, No. 1 (feb). p. 65-72. ISSN: 1988–3455.

PRESSMAN, Roger (2010). Ingeniería de software un enfoque práctico. 4 ed. México D.F. (México): McGraw - Hill. 640 p. ISBN: 8448132149.

QUIROGA, Jorge (2007). Gestión de conocimiento en grupos de desarrollo de software. En: Paradigma, Vol. 1, No 1 (abr). Bogotá (Colombia): Universidad de los Andes. p. 1-14. ISSN: 2011-0065.

RODRÍGUEZ, Mario; ARANZAZU, José; MURCIA, Francisco & ECHEVERRI, Carlos (2007). Aplicación de TSP en la Construcción de una Solución de Arquitectura Empresarial. En: Paradigma. Bogotá (Colombia): Universidad de los Andes. p. 1-16. ISSN: 2011-0065.

SOMMERVILLE, Ian (2011). Ingeniería de software. 7 ed. México D.F. (México): Person Education. 712 p. ISBN: 8478290745.

TORO, Fernando (1992). Estructura del instrumento, Cuestionario de motivación para el trabajo - CMT. En: Revista interamericana de Psicología Ocupacional. Medellín (Colombia): Centro de Investigación en Comportamiento Organizacional, CINCEL. ISSN: 0120-3800.