Evaluación de metodologías de desarrollo de Software utilizadas para valorar el impacto de un proyecto de software, durante un semestre de clases.

Marlon Jose Cardenas Castellanos
Facultad de Informática y Computación
Universidad Autónoma De Bucaramanga – Unab.
Maestría en Software Libre.
mcardenas574@unab.edu.co

Resumen.

La industria del software tiene grandes retos para el desarrollo de proyectos orientados a la eficiencia del producto final en términos de calidad.
Para esto se han generado diversas metodologías, procesos, normas y modelos que permiten afrontar estos retos, siendo no apropiado afirmar que la aplicación de una metodología específica logre cumplir con éxito un proyecto, ya que existe gran variedad de requerimientos con diferentes propósitos, clientes y mercados. Es por esto que se puede decir que la clave del éxito está en la correcta elección de las metodologías más apropiadas y su correcta integración al contexto organizacional.
La presente investigación contiene la integración de varias metodologías y normas para el proceso de desarrollo de software de una ambiente académico desde dos enfoques metodológicos tradicionales y agiles, en los cuales se exponen las principales modelos Cascada Rup, XP, Scrum, características, estructura, proceso, principios, ciclo de vida, artefactos y roles entre otras características propias de cada metodología.
Además se adicionó al proceso la evaluación del producto, a través de una norma de calidad de producto reconocida a nivel mundial como lo es la ISO 25000.

Abstract

The software industry has great challenges for the development of projects oriented to the efficiency of the final product in terms of quality.
To this end, a number of methodologies, processes, norms and models have been generated to address these challenges, and it is not appropriate to state that the application of a specific methodology successfully completes a project, since there is a great variety of requirements for different purposes, clients and Markets. That is why it can be said that the key to success lies in the correct choice of the most appropriate methodologies and their correct integration to the organizational context.
The present research contains the integration of several methodologies and norms for the software development process of an academic environment from two traditional and agile methodological approaches, in which the main models Cascada Rup, XP, Scrum, characteristics, structure, process , Principles, life cycle, artifacts and roles among other characteristics of each methodology.
In addition, product evaluation was added to the process through a globally recognized product quality standard such as ISO 25000.

Palabra Clave: Metodología, Proyecto de Software, Cascada Rup, XP, Scrum, ISO 25000

I. INTRODUCCIÓN.

En los proyectos de desarrollo de software no existe una única metodología para sacar adelante un proyecto, ya que el software es un producto intangible y difícil de medir en resultados, en este sentido, han surgido metodologías para minimizar este tipo de inconvenientes por medio de la utilización del uso frecuente de puntos de revisiones periódicas en el ámbito de calidad, adicionalmente una comunicación constante entre el usuario y los equipos de desarrollo del producto, permiten responder a las distintas necesidades de cliente de una forma más efectiva.

Existen investigaciones las cuales evidencian el uso de modelos tradicional para la administración de proyectos de software los cuales son poco eficiente, donde se asume que la construcción del software se puede dar en proceso continuo y lineal. Las metodologías ágiles, fueron creadas para eliminar estos problemas, implementando un modelo de forma colaborativa donde se interactúan con el cliente. [1].
Por este motivo la elaboración de esta investigación cobra importancia, ya que pretende involucrar las mejores técnicas, herramientas, procesos y factores de calidad de la metodología tradicional o agiles, para así obtener lo mejor con el marco de referencia la gerencia de proyectos.

II. MARCO TEÓRICO.

En la presente investigación describiremos los principales conceptos relacionados a un proyecto de desarrollo de software, iniciamos definiendo el concepto de metodologías de desarrollo de software y especificaremos las que fueron investigadas durante el proyecto. Asimismo, trabajaremos el concepto de calidad de software en especial la norma ISO/IEC 25000, enfocándonos a la eficiencia de una solución.

III. METODOLOGÍAS DE LA INDUSTRIA DEL DESARROLLO DE SOFTWARE.

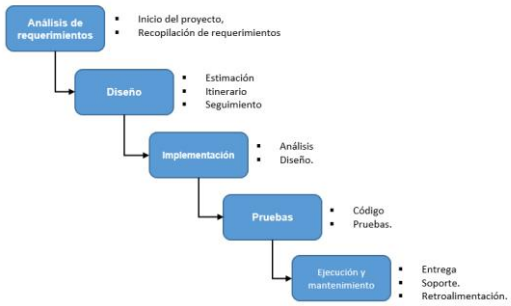
Una metodología de desarrollo de software se puede expresar como un conjunto de políticas, procesos y procedimientos que forman parte de un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo de software. [2]. La selección de una metodología de desarrollo varía dependiendo de los objetivos, tiempo y recursos con los que cuenta el equipo de desarrollo, tomando en cuenta estas características, desde 1970 se tienen antecedentes de los primeros modelos estandarizados para todo un proceso de desarrollo.
Analizando las características principales de las principales metodologías utilizadas en la industria, podemos referirnos a ellas en dos grupos diferentes: Metodologías tradicionales y Metodologías Ágiles. Para las metodologías tradicionales tenemos los primeros modelos que ganaron popularidad por ser estructurados y centrarse principalmente en la planificación como lo son Cascada, Espiral, Prototipado, RUP (Rational Unified Procces) y MSF (Microsoft Solution Framework).
Respecto a las metodologías ágiles podemos definir que principalmente se busca un modelo incremental con pequeñas entregas, una menor documentación y una actitud cooperativa entre cliente y desarrollador. Dentro de estas metodologías encontramos el Modelo SCRUM, LEAN, XP (Extreme Programming), ASD (Adaptative Software Development) y Crystal Clear [3].
Dentro de esta investigación nos centraremos en el estudio de dos de las metodologías, más usadas por los grupos de desarrollo de software: Modelo en Tradicional Cascada, RUP, y Modelo XP, Scrum. Como se explica más adelante, estas dos metodologías cuentan con ciertos lineamientos estructurados en las que posiblemente se pueda implementar conceptos de Usabilidad propios de nuevas metodologías.

IV. METODOLOGÍAS TRADICIONALES.

Las metodologías tradicionales de desarrollo de software son fundamentadas en la planeación de proyectos. Inician el desarrollo de un proyecto con un estricto proceso de secuencias de requerimientos, previo a las etapas de análisis y diseño, buscando lograr resultados con alta calidad ceñidos a un cronograma.
En las metodologías tradicionales se piensa en un solo proyecto, de grandes dimensiones y estructura definida; se sigue un proceso secuencial en una sola dirección y sin marcha atrás; el proceso es rígido y no cambia; los requerimientos son acordados de una vez y para todo el proyecto, demandando grandes plazos de planeación previa y poca comunicación con el cliente una vez ha terminado ésta. [4]

A. Modelo Cascada.

Este modelo identifica y demuestra mejor, la metodología de desarrollo tradicional, además de haber sido probado con el tiempo y ser de fácil comprensión. Este es un modelo estático y se caracteriza por gestionar los desarrollos de sistemas de una manera lineal y secuencial, completando una actividad antes que la otra, de allí proviene su nombre. Se compone de las siguientes 5 fases: Análisis de requerimientos, diseño, implementación (codificación), pruebas y ejecución y mantenimiento. (Ilustrado en la figura No 1.)
Fig 1. Modelo en Cascada.



(Pressman, 1995)

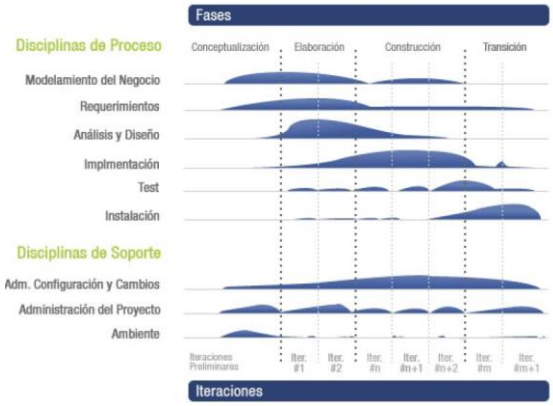
Como podemos identificar en la figura anterior, cuando una fase es realizada por completo se continuar con la siguiente, sin la posibilidad de regresar a una fase anterior, lo que requiere indispensablemente que el proyecto tenga un enfoque estructurado y orientado a los procesos. Sin embargo, esta característica hace que tenga una debilidad bien notoria, ya que si por alguna razón durante la fase de diseño surge algún problema que requiera regresar a la parte de análisis, o si el cliente solicita agregar o cambiar algunos requerimientos es imposible de realizar con el enfoque que tiene esta metodología.
Este modelo es muy útil en desarrollo de sistemas estructurados donde las modificaciones del software después de la codificación están prohibida. También, los datos y los procesos están usualmente separados, de manera que si se tiene que modificar la data, el código también tiene que ser modificado. Esto hace que el software no sea reutilizable y que el sistema sea difícil de actualizar, ya que si se desea hacer algún ajuste se tiene que modificar el proceso completo lo que puede ser muy complicado y costoso. [5]

B. Rational Unified Process (RUP).

RUP es una metodología que ofrece una perspectiva estructurada y disciplinada para asignar las tareas y recursos dentro de la organización. Su objetivo es asegurar que la producción de software, sea de alta calidad y que satisfaga las necesidades de sus usuarios, dentro de un cronograma y presupuesto establecido.
Una de las ventajas de usar RUP es que mejora la producción ya que proporciona a cada integrante del equipo un acceso a bases de conocimientos con información, guías, plantillas y herramientas para cualquier actividad crítica de desarrollo. De esta manera, al tener una base de conocimiento común no importa en qué fase trabaje o que rol tenga en el proyecto, todos los miembros del equipo compartirán una visión, procesos y objetivos en común de cómo desarrollar el software [6].

El Rational Unified Process proporciona a cada miembro del equipo las directrices, plantillas y herramientas necesarias para sacar el máximo provecho de las siguientes buenas prácticas:
1. Desarrollo de software iterativamente
2. Gestión de requerimientos
3. Uso de arquitecturas basada en componentes
4. Modelos de software visual
5. Verificación de la calidad del software
6. Control de cambios del software.

Fig. 2. Estructura de RUP**.**



Adaptado de RUP (IBM)

RUP se divide en cuatro fases que son: Incepción, Elaboración, Construcción y Transición. Estas fases se dividen en iteraciones, donde cada una tiene el propósito de producir una parte demostrable del software (Artefacto). La duración de cada iteración varía desde dos semanas hasta seis meses.
En la fase de incepción, se establecen todos los objetivos del proyecto considerando todas las necesidades de todos los involucrados en el proyecto (usuarios finales, compradores o contratistas). Esto conlleva a establecer artefactos como el alcance, condiciones de entorno y criterios de aceptación del proyecto. Los casos de uso críticos son identificados, el cronograma y los costos son estimados y se comienzan a plantear posibles arquitecturas para el sistema.
En la fase de elaboración es donde se sientan las bases de la arquitectura de software. En esta etapa se analiza el problema, se realiza el plan de proyecto, se describen en detalle el proceso, la infraestructura y el entorno de desarrollo y se identifican los actores y los casos de uso. Al final de esta fase, se realiza un análisis para determinar los riesgos, la estabilidad de la visión y de la arquitectura y el gasto de los recursos en comparación con lo que se había planeado inicialmente.
En la fase de construcción, todos los componentes y funcionalidades de la aplicación son desarrollados e integrados en el producto. RUP considera esta fase

como un proceso de elaboración donde se pone énfasis en la gestión de los recursos, control de costos, de cronograma y calidad. Los resultados de la fase de construcción se crean lo más rápido posible sin dejar de lograr una calidad adecuada. Se llega a la fase de transición cuando el producto software se encuentra lo suficientemente maduro para poder ser entregado a los usuarios. Con la retroalimentación del cliente se procede a corregir los problemas o culminar alguna característica pendiente. En esta fase también se realizan las pruebas beta, pilotos, la capacitación de los usuarios y administradores del sistema [5]

V. METODOLOGÍAS ÁGILES.

La evolución de las TIC y los entornos donde se desarrollan los negocios hicieron que las metodologías tradicionales asumieran el reto de adaptarse a los cambios. Por lo cual, poco pudieron hacer para asumir de manera eficaz a estas variantes, ya que no tenían la capacidad de responder de manera rápida a los cambios que se daban en todo el ciclo de desarrollo y que determinaban en casi todos los casos el fracaso del producto final. Debido a esto, surgieron un nuevo tipo de metodologías que estaban basadas en el conocimiento de que los requerimientos eran cambiantes y dinámicos, las metodologías de desarrollo ágil [7].

El enfoque de desarrollo ágil se basa en la idea de un desarrollo iterativo incremental, donde cada fase dentro del ciclo de vida de desarrollo es revisado varias veces, lo que permite que se realice mejoras continuas en base a las retroalimentaciones que realice el cliente con respecto al avance que se le va presentando periódicamente. En el enfoque ágil, en lugar de trabajar con un único y enorme modelo de proceso que se implementa en los enfoques tradicionales, el desarrollo del ciclo de vida es divido en pequeñas partes más pequeñas llamadas iteraciones, donde cada una de estas partes tienen definido un objetivo en específico y forma parte de una de las fases del desarrollo convencional [8].

De acuerdo al Manifestó Ágil los principales factores del enfoque ágil son los siguientes:

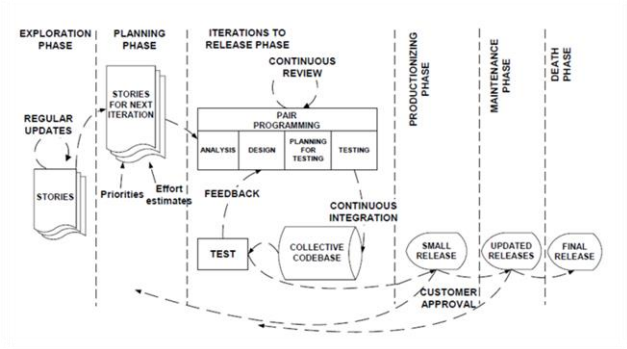
1.- La participación temprana del cliente.
2.- El desarrollo iterativo
3.- Equipo auto-organizados
4.- La adaptación al cambio.

A continuación se detallaran dos de las metodologías ágiles más conocidas y que fueron referenciadas durante la investigación realizada para este proyecto de tesis. Estas son: SCRUM y Extreme Programming.

A. Programación Extrema XP.

La programación extrema o Extreme Programming, es una disciplina de desarrollo de software basada en los métodos ágiles, que evidencia principios tales como el desarrollo incremental, la participación activa del cliente, el interés en las personas y no en los procesos como elemento principal, y aceptar el cambio y la simplicidad [9]. Su éxito se debe al enfoque en la satisfacción del cliente, ya que ofrece el software que se necesite en el momento que se requiere. Esto se debe a la capacitación de sus desarrolladores para poder responder de la manera más eficiente a los cambios en los requerimientos del cliente, aun cuando estos se den en las últimas etapas del ciclo de vida, la cual consiste en 5 fases: Exploración, planificación, iteraciones para despliegue, pases a producción, mantenimiento y muerte. Estas se presentan en el siguiente gráfico:

Fig. 3. El proceso XP.



Adaptado de *[3]*.

En la fase de exploración los clientes escriben los requerimientos que deben ser agregados al programa, en una historia de usuario para una versión en específico. Paralelamente, el equipo de proyecto adecua las herramientas, tecnologías y prácticas que van a ser usadas, mientras que las tecnologías y posibles arquitecturas son probadas y exploradas mediante la creación de prototipos del sistema. Esta fase toma aproximadamente entre unas semanas o pocos meses dependiendo de cuán familiarizados están los programadores a esta nueva tecnología.

En la fase de planificación se realiza una priorización de todas las historias de usuario y se realiza un acuerdo de todo el contenido destinado para la primera versión. También se realiza un cronograma en base a la estimación que realizan los programadores para determinar cuánto esfuerzo requiere implementar cada historia de usuario. El periodo de tiempo del cronograma normalmente no excede de los dos meses.

En la fase de iteraciones para el despliegue se procede a descomponer el calendario establecido en la fase

anterior en un conjunto de iteraciones que tomarán entre una y cuatro semanas implementar cada una. La primera iteración crea un sistema con la arquitectura completa, luego el cliente selecciona las historias de usuario para cada iteración y crea las pruebas funcionales que van a ser ejecutadas al final de cada iteración. Al final de la última iteración el producto está listo para pasar a producción. [10].

La fase de pases a producción demanda que se realicen pruebas adicionales y comprobaciones del rendimiento del producto antes de que sea entregado al cliente. Durante esta fase aún es posible encontrar nuevos cambios y se tiene que tomar una decisión si es que esos cambios van a ser incluidos en la entrega. Las ideas postergadas y las sugerencias son documentadas para una futura implementación.

Luego de que la primera versión pasara a producción para el uso de cliente, el proyecto en XP entra en la fase de mantenimientos donde debe mantener tanto el sistema corriendo en producción como el desarrollo de nuevas iteraciones. Debido a esto, la velocidad de desarrollo tiende a desacelerarse y se ve necesario incorporar nuevos miembros y cambiar la estructura del equipo para que el proyecto no se vea afectado.

Por último, se sabe que se está llegando a la fase de la muerte cuando el cliente ya no tiene ninguna historia de usuario que implementar. Este es el momento donde se puede escribir en la documentación del proceso que no se van a realizar ningún cambio más en la arquitectura, diseño o codificación del proyecto. Esta etapa también puede ocurrir si el sistema no entrega los resultados esperados o si se vuelve demasiado caro para un futuro desarrollo.
[11].

B. Scrum.

Este modelo es un framework, estructurado para soportar el desarrollo de productos de software complejos. Se compone de Scrum Teams y sus roles, eventos, artefactos y reglas asociadas. Cada componente dentro del framework tiene un propósito en específico y es esencial para su éxito y uso [12].

El enfoque de scrum ha sido desarrollado para la gestión del proceso de desarrollo de sistemas. Esta metodología no tiene definido ninguna técnica de desarrollo de software para la fase de implementación, ya que se enfoca en cómo los miembros del equipo deben funcionar para poder producir el sistema de manera flexible en un ambiente de constante cambio.

La idea principal de Scrum es que el desarrollo de sistemas involucra variables de entorno y técnicas que están propensas a cambiar durante el proceso. Es por esto, que el desarrollo se vuelve muy impredecible y complejo y demanda que el proceso de desarrollo tenga la capacidad y flexibilidad suficiente para poder hacer frente a estos cambios constantes.

El proceso de scrum se divide en tres fases: pre-game, desarrollo y post-game, como se muestra en la siguiente ilustración.

Fig 4. proceso Scrum:

(Abrahamsson, 2002).

En la fase pre-game tenemos dos sub faces que lo componen: Planificación y la arquitectura. En la etapa de planificación incluye la toma de requerimientos del sistema que se está desarrollando. Estos requerimientos pueden venir de diferentes fuentes (cliente, vendedores, división de marketing, soporte del cliente o de desarrolladores de software), todos estos requerimientos son listados y priorizados en el Product Backlog donde se estima el esfuerzo necesitado para poder implementar cada uno. En esta etapa también incluye la definición del equipo de proyecto, las herramientas a usar y otros recursos, la evaluación de riesgos y control, entrenamiento necesario y las aprobaciones por parte de la gerencia. En cada iteración el Product Backlog es revisado por el Scrum Team para ganar su aprobación y compromiso para empezar la siguiente iteración.

En la fase de arquitectura se planea el diseño de alto nivel y la arquitectura del sistema de acuerdo a los elementos que se tienen en el Product Backlog. En esta etapa se requiere una reunión donde se revisa las propuestas para la implementación y se toma una decisión a partir de esta revisión.

La parte de desarrollo es tratada como una caja negra, donde lo impredecible puede ocurrir. Es en esta etapa donde las diferentes variables de entorno y técnicas (como el marco, calidad, requerimientos y herramientas) que pueden cambiar en cualquier momento durante el proceso de desarrollo son observadas y controladas por varias prácticas de Scrum durante los Sprints para poder ser capaces de adaptarse con flexibilidad ante cualquier cambio que se afronte. En esta etapa el software es desarrollado en Sprints, ciclos iterativos donde una funcionalidad es desarrollada o mejorada para producir nuevos avances. Cada Sprint incluye las fases tradicionales del desarrollo de software (requerimientos, análisis, diseño, evolución y entrega) y planeada para durar entre 1 semana a un mes.

La fase de post-game contiene el cierre del producto. Esta fase se da cuando se ha llegado a un acuerdo de que los requerimientos solicitados han sido completados. En esta etapa se realiza todas las preparaciones para poder pasar a producción, tareas como integración, pruebas del software y la documentación. [7].

VI. CALIDAD DE SOFTWARE.

La calidad del software es la correlación con los requerimientos funcionales cumpliendo con las políticas y el rendimiento establecidos por el proyecto. Desarrollar un software con calidad implica la utilización de estándares, metodologías y procesos para análisis, diseño, programación y pruebas, con el fin de lograr confiabilidad, efectividad y productividad para el control de la calidad del software. [13].
Actualmente en el ámbito académico ven la necesidad no solo de entregar sus productos según los requerimientos, sino con características y atributos de calidad, por tanto estos proyectos tienen implementado en sus procesos el área de QA en donde la satisfacción del usuario y un producto confiable son la prioridad.

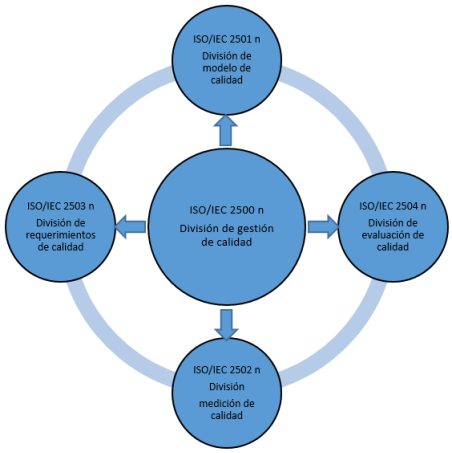
A Norma ISO/IEC 25000.

Las normas ISO/IEC 25000 también llamadas SQuare (Requisitos y Evaluación de Calidad de Productos de Software) están conformadas por las normas ISO/IEC 9126 e ISO/IEC 14598, surgen para crear modelos, métricas, procesos y herramientas de evaluación de calidad del software como producto, por medio de la especificación de los requisitos.
Es una familia de normas que tiene por objetivo la creación de un marco de trabajo común para evaluar la calidad del producto software [14]

B. Estructura de la familia ISO/IEC 25000

La ISO/IEC 25000 se encuentra compuesta por varias divisiones, entre las que se destacan las siguientes figura 5:

Fig. 5.Modelo divisiones norma ISO/IEC 25000.

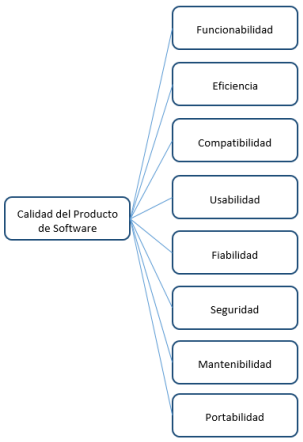


Norma ISO/IEC 25000.

B. Modelo de Calidad

A la hora de establecer la calidad de un producto de software es importante definir un modelo que permita realizar una evaluación detallada con una secuencia específica, que además permita estructurar los puntos a evaluar.
La norma ISO 2501n presenta un modelo de calidad detallado donde incluye las características de calidad interna, externa y para la calidad en uso, y está formada por:
ISO/IEC 25010 Modelos del sistema y calidad del software: Detalla el modelo de la calidad del producto, describiendo ocho características para evaluar el software, las cuales son:
A continuación en la figura 6, se detallan algunas de las características del modelo de calidad

Figura 6. Características calidad interna y externa del producto software.



Fuente ISO/IEC 25000.

VII. CONCLUSIONES.

Podemos concluir en el ejercicio de la investigaciones obtuvo información importante para ser aplicada en la catedra Ingeniería de Software y en la administración de proyectos de desarrollo de la Universidad de Cartagena, la cual puede ser utilizada para impartir una mejor formación a los estudiantes e impulsar los semilleros de desarrollo.
A continuación se realiza la discusión de los principales resultados obtenidos, su validez, los alcances y algunas sugerencias para continuar con otras investigaciones

La investigación permitió obtener información importante para determinar las prácticas y metodologías, en proyectos de desarrollo de software orientados a la eficiencia y el desempeño de estos.
Con el fin de discutir los hallazgos y las observaciones realizadas hemos dividido en las tres siguiente secciones.

A. Identificar prácticas docentes en el desarrollo de proyectos de software.

Ambas metodologías tienen sus limitaciones y debilidades, así como las metodologías ágiles son las más adecuadas para proyectos pequeños y medianos, no son las más adecuadas para sistemas de gran escala que requieran de interacciones complejas con otros sistemas, debido a que estos sistemas requieren de un nivel de precisión bastante alto y tienen un gran riesgo de construcción.
No sería conveniente implementar una metodología ágil para el desarrollo de un sistema crítico en el cual es necesario el análisis detallado de todos los requerimientos para comprender su complejidad e implicaciones, debido a la complejidad y la extrema precisión que pueda tener la captura de requerimientos, en los cuáles las metodologías agiles como SCRUM ofrecen demasiada flexibilidad.
(Karlstrm y Runeson, 2010) encontraron que los métodos agiles proveen herramientas ponderosas para la planeación a pequeña escala, control del trabajo diario, reporte de progreso y la mejora en los canales de comunicación del equipo, evidenciando la tendencia a las metodologías agiles a grupos de desarrollo de la Universidad de Cartagena, que requerían mucha interacción con el cliente y constantes variaciones en los requerimientos del proyecto, mientas en otras investigaciones aplicadas al campo de desarrollo [4], se evidencio la utilización del modelo XP.

Por otro lado, la teoría hace ver RUP como un proceso rígido e inmodificable por la gran cantidad de documentación que es necesaria completar antes de la finalización de cada etapa, pero realmente es un proceso de software configurable, donde a cada stakeholder se le asignan sus actividades acorde con su rol dentro del proyecto, además del uso de una definición común para el proceso que es compartida por todo el equipo de desarrollo, y de esta forma asegurar una comunicación clara y sin ambigüedades entre los miembros del equipo, así mismo establecimos que un 70% de los grupo de desarrollo utilizan a Rup como el modelo ideal al aplicar metodologías tradicionales y es la metodología preferida para el proceso de enseñas en la asignatura ingeniería de software.

B. Criterios de medición para determinar la eficiencia de las metodologías.

Se aplicó la matriz de evaluación de metodologías es imprescindible que la persona que va a realizar la ponderación conozca a fondo las metodologías a evaluar, ya que esto podría traer como consecuencia no seleccionar la que en realidad es la más conveniente para el tipo de proyecto a desarrollar, por ende el tutor de la formación colocara las ponderaciones a cada metodología evaluada, para las metodologías tradicionales Rup 70 %, obtuvo la más alta puntuación mientras en las metodologías agiles Scrum obtuvo el más alto promedio 80 %.

C. Niveles de desempeño según criterios de calidad.

En el criterio de calidad tomamos de la norma unas características y subcaracterísticas; sin embargo, sugerimos modificar el modelo para adaptarlo a las necesidades de cada proyecto o grupo desarrollador. Es decir, tomar o adaptar las que sean útiles para los objetivos de calidad que se desean alcanzar.
Las investigaciones realizadas en el ámbito académico pueden ser llevadas al ambiente empresarial; para lo cual, las empresas de software deberían definir las principales características y subcaracterísticas que desean evaluar en los productos que desarrollan. Esto permitiría alcanzar los objetivos de calidad deseados y optimizar los recursos existentes,

sean estos económicos, recursos humanos y tiempo. Se recomendaría que las medidas tengan escalas como: el mejor posible, peor posible y el resultado deseable, o sino también el mínimo aceptable.

En el desempeño se tiene presente las subcaracteristicas de la eficiencia como son: comportamiento del tiempo, utilización de recursos y cumplimiento de eficiencia, donde el 90% de los proyectos se reflejaba un alto porcentaje en el cumplimiento en los diferentes proyectos, evidenciando que la metodología y estándar de calidad de software a utilizar, dependerá de los objetivos que se pretende alcanzar.

En otras investigaciones relacionadas a calidad de software [5], se identifican las características a evaluar según la norma ISO/IEC 25000, referenciando los objetivos del proyecto y la utilización de metodologías agiles para afrontar proyectos con acompañamiento del cliente y aunque sea utilizada metodologías tradicionales o agiles, se debe evaluar la calidad del software basado en normas y con las misma rigurosidad.

RECONOCIMIENTOS

Agradeciendo a la Universidad de Cartagena por permitir desarrollar la investigación, en las aulas y laboratorios de desarrollo de software.

REFERENCIAS.

[1] The Standish Group, «CHAOS Report,» Grupo Standish , Estados Unidos, 2012.

[2] R. Pressman y J. Murrieta, Ingeniería delsoftware un enfoque práctico, Madrid: McGraw-Hill, 2010.

[3] R. G. Figueroa, C. J. Solís y A. A. Cabrera , «Metodologías Tradicionales vs. Metodologías Ágiles, Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación,» 2008. [En línea]. Available: http://adonisnet. files. wordpress. com/2008/06/articulo-metodologia-de-sw-formato. doc.

[4] A. Navarro Cadavid, J. D. Fernández Martínez y J. Morales Vélez, «Revisión de metodologías ágiles para el desarrollo de software,» *Prospectiva, ISSN-e 2216-1368, Vol. 11, Nº. 2 (julio-diciembre),* p. 30-39, 2013.

[5] F. Amo, L. Martínez y F. Segovia, Introducción a la ingeniería del software: Modelos de desarrollo de programas, Madrid (España): Delta Publicaciones. p 335-349, 2005.

[6] IBM, «Rational Software Corporation,» 1998. [En línea]. Available: http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf.

[7] M. Alaimo y M. Salias, Proyectos Agiles con Scrum, Buenos Aires: Kleer, 2015.

[8] L. Merchán Paredes, A. Urrea y R. Rebollar, «Definicion de una metodologia agil de ingenieria de requerimientos para empresas emergentes de desarrollo de Software del sur-occidente de Colombia,» *Guillermo de Ockham: Revista científica, ISSN 1794-192X, Vol. 6, Nº. 1,* pp. 37-50, 2008.

[9] K. Beck y C. Andres, Extreme Programming Explained, Addison Wesley, Stoughton.: Embrace Change [2ª ed.], 2004.

[10] D. Wells, «Extreme Programming: A gentle introduction,» 18 Febrero 2011. [En línea]. Available: http://www.extremeprogramming.org/.

[11] P. Abrahamsson, O. Salo, J. Ronkainen y J. Warsta, Agile software development methods: Review and analysis, VTT Publications, 2002.

[12] S. Mariño y P. L. Alfonzo, «Implementación de SCRUM en el

diseño del proyecto del Trabajo Final de Aplicación,» *Scientia et Technica, ISSN 0122-1701, Vol. 19, Nº. 4,* p. 413-418, 2014.

[13] R. S. Pressman, «Ingenieroa del Software un enfoque practico,» McGrawhill, España, 1998.

[14] J. Llorens, Gerencia de proyectos de tecnología de información, Caracas: El Nacional, 2005.

[15] E. M. Mendez Nava, Artist, *Modelo de Evaluacion de Metodologias para el Desarrollo de Software*. [Art]. Universidad Catolica Andres Bello, 2006.

[16] P. Roa Molina y C. Morales, «Norma de calidad ISO/IEC 25000 en desarrollo de Software,» *TIA*, p. 7, 2015.