UNIVERSIDAD ESTATAL DE MILAGRO

FACULTAD CIENCIAS DE LA INGENIERIA

TRABAJO DE TITULACIÓN DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERIA EN SISTEMAS COMPUTACIONALES

PROPUESTA PRÁCTICA DEL EXAMEN COMPLEXIVO

TEMA: METODOLOGÍAS BASADOS EN PROCESOS DE DESARROLLO DE SOFTWARE: ANÁLISIS ENTRE LOS ENFOQUES METODOLÓGICOS FORMALES VS. ENFOQUES METODOLÓGICOS ÁGILES DE DESARROLLO

Autores:

OLMEDO MERELO NESTOR STEVEN

TENORIO ALMACHE JESSICA FABIOLA

Acompañante:

MSc. Lissett Margarita Arévalo Gamboa

Milagro, agosto 2017

ECUADOR

DERECHOS DE AUTOR

Ingeniero.
Fabricio Guevara Viejó, PhD.
RECTOR
Universidad Estatal de Milagro
Presente.

Yo/Nosotros, **Olmedo Merelo Néstor Steven, Tenorio Almache Jessica Fabiola** en calidad de autor(es) y titulares de los derechos morales y patrimoniales de la propuesta práctica de la alternativa de Titulación - Examen Complexivo, modalidad presencial, mediante el presente documento, libre y voluntariamente procedo a hacer entrega de la Cesión de Derecho del Autor de la propuesta practica realizado como requisito previo para la obtención de mi (nuestro) Título de Grado, como aporte a la Temática "**: Metodología de los Procesos de Desarrollo de Software: Análisis entre los Enfoques metodológicos formales vs. Enfoques metodológicos ágiles de desarrollo**," del Grupo de Investigación **TIC Y Desarrollo de Software** de conformidad con el Art. 114 del Código Orgánico de la Economía Social De Los Conocimientos, Creatividad E Innovación, concedemos a favor de la Universidad Estatal de Milagro una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Conservamos a mi/nuestro favor todos los derechos de autor sobre la obra, establecidos en la normativa citada.

Así mismo, autorizo/autorizamos a la Universidad Estatal de Milagro para que realice la digitalización y publicación de esta propuesta practica en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

El (los) autor (es) declara (n) que la obra objeto de la presente autorización es original en su forma de expresión y no infringe el derecho de autor de terceros, asumiendo la responsabilidad por cualquier reclamación que pudiera presentarse por esta causa y liberando a la Universidad de toda responsabilidad.

Milagro, a los 21 días del mes de septiembre del 2017

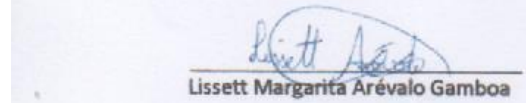
Firma del Estudiante (a)
Nombre: Jessica Fabiola Tenorio Almache
CI: 0940367295

Firma del Estudiante (a)
Nombre: Néstor Steven Olmedo Merelo
CI: 0942091083

APROBACIÓN DEL ACOMPAÑANTE DE LA PROPUESTA PRÁCTICA

Yo, **Lissett Margarita Arévalo Gamboa** en mi calidad de acompañante de la propuesta práctica del Examen Complexivo, modalidad presencial, elaborado por los estudiantes **Olmedo Merelo Néstor Steven, Tenorio Almache Jessica Fabiola**; cuyo tema es: **Metodología de los Procesos de Desarrollo de Software: Análisis entre los Enfoques metodológicos formales vs. Enfoques metodológicos ágiles de desarrollo**, que aporta a la Línea de Investigación **Tecnologías de la Información y de la Comunicación** previo a la obtención del Grado de **Ingenieros en Sistemas Computacionales**; considero que el mismo reúne los requisitos y méritos necesarios en el campo metodológico y epistemológico, para ser sometido a la evaluación por parte del tribunal calificador que se designe, por lo que lo APRUEBO, a fin de que el trabajo sea habilitado para continuar con el proceso de titulación de la alternativa de Examen Complexivo de la Universidad Estatal de Milagro.

En la ciudad de Milagro, a los 21 días del mes de septiembre de 2017.



Lissett Margarita Arévalo Gamboa

ACOMPAÑANTE
CC. 0925716987

APROBACIÓN DEL TRIBUNAL CALIFICADOR

El tribunal calificador constituido por: **Arévalo Gamboa Margarita Lissett, Correa Peralta Mirella Azucena, Córdova Martínez Luis Cristóbal.**

Luego de realizar la revisión de la propuesta práctica del Examen Complexivo, previo a la obtención del título (o grado académico) de **Ingeniería en Sistemas Computacionales** presentado por el (la) señor (a/ita) **Olmedo Merelo Néstor Steven**

Con el título:

Metodologías basados en procesos de desarrollo de software: análisis entre los enfoques metodológicos formales vs. Enfoques metodológicos ágiles de desarrollo

Otorga al presente la propuesta práctica del Examen Complexivo, las siguientes calificaciones:

MEMORIA CIENTÍFICA	[95]
DEFENSA ORAL	[5]
TOTAL	[100]
EQUIVALENTE	[50]

Emite el siguiente veredicto: (aprobado/reprobado) Aprobado

Fecha: 21 de Septiembre del 2017.

Para constancia de lo actuado firman:

	Nombres y Apellidos	Firma
Presidente	Lissett Arévalo Gamboa	
Vocal 1	LUIS CORDOVA MARTINEZ	
Vocal 2		

APROBACIÓN DEL TRIBUNAL CALIFICADOR

El tribunal calificador constituido por: **Arévalo Gamboa Margarita Lissett, Correa Peralta Mirella Azucena, Córdova Martínez Luis Cristóbal.**

Luego de realizar la revisión de la propuesta práctica del Examen Complexivo, previo a la obtención del título (o grado académico) de **Ingeniería en Sistemas Computacionales** presentado por el (la) señor (a/ita) **Tenorio Almache Jessica Fabiola**

Con el título:

Metodologías basados en procesos de desarrollo de software: análisis entre los enfoques metodológicos formales vs. Enfoques metodológicos ágiles de desarrollo.

Otorga al presente la propuesta práctica del Examen Complexivo, las siguientes calificaciones:

MEMORIA CIENTÍFICA	[95]
DEFENSA ORAL	[5]
TOTAL	[100]
EQUIVALENTE	[50]

Emite el siguiente veredicto: (aprobado/reprobado) Aprobado

Fecha: 21 de Septiembre del 2017.

Para constancia de lo actuado firman:

	Nombres y Apellidos	Firma
Presidente	Lissett Arévalo Gamboa	
Vocal 1	LUIS CRISTÓBAL CÓRDOVA MARTÍNEZ	
Vocal 2	Mirella Correa Peralta	

Contenido

DEDICATORIA

A Dios por permitirme llegar a este momento tan especial en mi vida, por los triunfos y los momentos difíciles que me han enseñado a valorarte cada día más. A ti Madre por haberme educado y soportar mis errores, gracias a tus consejos, por el amor que siempre me has brindado, por cultivar e inculcar ese sabio don de la responsabilidad. ¡Gracias por darme la vida! ¡Te amo mucho! A ti Padre, a quien le debo todo en la vida, le agradezco el cariño, la comprensión, la paciencia y el apoyo que me brindó para culminar mi carrera profesional. A mis Hermanos Porque siempre he contado con ellos para todo, gracias a la confianza que siempre nos hemos tenido; por el apoyo y amistad ¡Gracias!, a todos mis familiares que me resulta muy difícil poder nombrarlos en tan poco espacio, sin embargo, ustedes saben quiénes son. A mis maestros. Gracias por su tiempo, por su apoyo, así como por la sabiduría que me transmitieron en el desarrollo de mi formación profesional, en especial: al Ing. Jorge Vinueza, Ing. Víctor Rea, Ing. Mirella Correa, Ing. Javier Martínez y Ing. Ángel. A mis amigos, que gracias al equipo que formamos logramos llegar hasta el final del camino y que, hasta el momento, seguimos siendo amigos: Luis Matute, Deyanira Onofre, Jonathan Villamar, Bryan Carbajal, Fabian Huaraca, Alexis Santacruz, y principalmente a mi gran amiga, novia y compañera de propuesta tecnológica Jessica Fabiola Tenorio. A la Universidad Estatal de Milagro y en especial a la Facultad de Ingeniería en Sistemas que me dieron la oportunidad de formar parte de ellas. ¡Gracias!

Néstor Steven Olmedo Merelo

Dedico este proyecto de tesis a Dios porque ha estado conmigo a cada paso que doy, cuidándome y dándome fortaleza para continuar y a mis queridos padres Segundo José Tenorio y Fabiola Almache por estar cuando más los necesito con sus consejos y amor de padres supieron guiarme por el buen camino de la vida formándome en una excelente mujer de provecho y quienes con su apoyo incondicional han hecho posible que mi formación intelectual y académica haya culminado con éxito. A mis hermanos Leonardo, José Luis y Diego quienes, con su cariño, respeto y depositaron su entera confianza en cada reto que se me presentaba sin dudar ni un solo momento en mi inteligencia y capacidad, dándome la fuerza y apoyo para seguir en mi carrera profesional. Y por supuesto a todos mis compañeros de clase que con sus ocurrencias y sabidurías hemos sabido aprovechar para apoyarnos y salir adelante en todo momento, en especial a mi amigo, novio y compañero de propuesta tecnológica.

Jessica Fabiola Tenorio Almache

AGRADECIMIENTO

Le agradezco a Dios por haberme acompañado y guiado a lo largo de mi carrera, por ser mi fortaleza en los momentos de debilidad y por brindarme una vida llena de aprendizajes, experiencias y sobre todo felicidad.

Le doy gracias a mis padres Gerardo Casco Arévalo y Aracely Merelo Sánchez por apoyarme en todo momento, por los valores que me han inculcado, y por haberme dado la oportunidad de tener una excelente educación en el transcurso de mi vida. Sobre todo, por ser un excelente ejemplo de vida a seguir.

A mis hermanos Joel y Odalys por ser parte importante de mi vida y representar la unidad familiar.

A Fabiola, por haber sido una excelente amiga, compañera de propuesta tecnológica, y por ser una parte muy importante de mi vida, por haberme apoyado en las buenas y en las malas, sobre todo por su paciencia y amor incondicional. T. A Mamut.

Le agradezco la confianza, apoyo y dedicación de tiempo a mis profesores: Javier Martínez por haberme incentivado a la línea de la investigación, Víctor Rea, Jorge Vinueza, Mirella Correa, Daniel Vera, Mariuxi Vinueza, Franklin Toaquiza, Margarita Arévalo y especialmente a Ángel Cuenca. Por haber compartido conmigo sus conocimientos y sobre todo su amistad.

A mis amigos por confiar y creer en mí y haber hecho de mi etapa universitaria un trayecto de vivencias que nunca olvidaré.

Néstor Steven Olmedo Merelo

En primer lugar, a Dios por haberme guiado por el camino de la felicidad; en segundo lugar, a cada uno de los que son parte de mi familia a mi padre José Tenorio, mi madre Fabiola Almache, y a mis hermanos; por siempre haberme dado su fuerza y apoyo incondicional que me han ayudado y llevado hasta donde estoy ahora.

A la Universidad Estatal de Milagro por habernos permitido formar parte de esta gran comunidad Universitaria.

A mis maestros Javier Martínez, Víctor Rea, Jorge Vinueza, Mirella Correa, Daniel Vera, Franklin Toaquiza, es especial a mi acompañante de dicha propuesta, quién nos ayudó en todo momento, MSc. Margarita Arévalo.

Y, por último, a mi amigo, novio y compañero de propuesta tecnológica porque en esta armonía grupal lo hemos logrado y culminado con éxito.

Jessica Fabiola Tenorio Almache

TEMA: "Metodologías basados en procesos de desarrollo de software: análisis entre los enfoques metodológicos Formales vs. Enfoques metodológicos ágiles de desarrollo"

RESUMEN

Actualmente existen diversas prácticas o métodos de trabajo, al momento de querer desarrollar un software, debido a que no es una tarea sencilla, pero siempre tratando de optimizar procesos, para que de esta forma los resultados sean más agiles y de mayor calidad, permitiéndose ser beneficiados tanto el cliente como los desarrolladores, sin embargo estas metodologías no del todo son perfectas ya que tienen sus ventajas y desventajas por lo que muchas son muy utilizadas y otras no, es por esta razón que tenemos que analizar con base al proyecto cual es la metodología más adecuada para llevar este proceso, anteriormente se podían encontrar metodologías estructuradas o también llamadas formales que consiste en el manejo del proceso disponiendo explícitamente las tareas involucradas, manifestando ser eficientes e indispensables en proyectos, pero a su vez han causado inconvenientes en otros. Cabe destacar que hoy en día se han vuelto muy populares otro tipo de metodología llamada ágiles, la filosofía de este método se basa en el factor humano es decir darle mayor valor al individuo con respecto a la comunicación frecuente con el cliente y al progreso incremental del software con repeticiones muy cortas, dando como resultado la mejor efectividad en diversos proyectos que contengan requisitos variantes y a su vez cuando se encuentran obligados a reducir el tiempo de desarrollo de una manera drástica, siempre y cuando conserven un alto nivel de calidad, debido que estas metodologías alteran la forma de crear programas estableciendo una amplia discusión por sus partidarios y algunos que por desconfiados no las desean como elección en comparación de las metodologías estructuradas o formales. Por lo tanto, la complejidad del desarrollo de sistemas de software es algo que siempre

va a estar presente, de lo contrario todas las empresas y clientes tendrían un software ideal, sin errores.

PALABRAS CLAVE:

Metodología ágil, Metodología tradicional, Metodología formal, Desarrollo de software, Método de desarrollo, Programación Extrema, RUP, XP, Proceso Unificado Racional.

TITTLE: "Methodologies based on software development processes: Analysis between formal methodological approaches VS Agile methodological approaches to development"

ABSTRACT

Currently there are various practices or methods of work, at the time of wanting to develop software, because it is not a simple task, but always trying to optimize processes, so that the results are more agile and higher quality, allowing to benefit both the client and the developers, however these methodologies are not completely perfect because they have their advantages and disadvantages, so many are very used and others are not, It is for this reason that we have to analyze based on the project which is the most adequate methodology to carry out this process, previously it was possible to find structured or also formal methodologies, before, structured or formal methodologies could found, which consists of the management of the process, explicitly arranging the tasks involved, claiming to be efficient and indispensable in projects, but in turn have caused inconveniences in others. Nowadays have become very popular another type of methodology called agile, the philosophy of this method is based on the human factor is to give

greater value to the individual with regard to frequent communication with the client and incremental progress Software with very short repetitions, resulting the best effectiveness in diverse projects that contain variant requirements and in turn when they are forced to reduce the time of development of a drastic way, as long as they maintain a high level of quality, because these methodologies alter the form to create Programs setting a wide discussion by their supporters and some who by distrust do not want them as an election in comparison to structured or formal methodologies.

Therefore, the complexity of developing software systems is something that will always be present, otherwise all companies and customers would have an ideal software, without errors.

KEYWORDS:

Agile Methodology, Traditional Methodology, Formal Methodology, Software Development, Development Method, Extreme Programming, RUP, XP, Rational Unified Process.

INTRODUCCIÓN

En inicios, los sistemas fueron softwares de amplitud reducida por causa del poco rendimiento que brindaba el hardware en aquel momento. Con el pasar del tiempo el desarrollo de la tecnología fue mejorando y con ello el rendimiento que brindaba el hardware, permitiendo que creciera la magnitud y dificultad en el tiempo del desarrollo del sistema**.** Por este motivo surge la necesidad de crear una metodología para el proceso de software.

Podemos definir, en el contenido que es un proceso que conlleva a una serie de actividades u operaciones permitiendo cumplir un objetivo específico. En mayoría las empresas y compañías se necesitan de uno o varios procesos para cumplir con sus propósitos, en el cual en alguno de ellos se necesitan sistemas de software. Cuando nos referimos a una empresa que se dedica a la comercialización de software necesita procesos que engloben desde que se crea el software hasta la finalización del mismo, motivo por el cual posee el nombre de "ciclo de vida del software".

Al momento de desarrollar un software no es una tarea fácil, por la cual en la actualidad existen diversas prácticas o métodos de trabajo que se han venido analizando en tiempos anteriores con el trabajo de otras personas para favorecer el desarrollo del software, siempre tratando de optimizar procesos, que los resultados sean más agiles y de mayor calidad permitiéndose ver beneficiados tanto el cliente como los desarrolladores, estas metodologías no son siempre perfectas tienen sus ventajas y desventajas por lo que muchas son muy utilizadas y otras no, básicamente no hay un método universal que se puede aplicar en todos los proyectos siempre va ver riesgos o desventajas por ello tenemos que analizar con base al proyecto cual es la metodología más adecuada. Con el pasar del tiempo los diversos métodos se han mejorado y actualizado, anteriormente se podían encontrar metodologías estructuradas o llamadas formales que se enfocan en el manejo del proceso disponiendo explícitamente las tareas involucradas,

estos métodos han manifestados ser eficientes e indispensables en proyectos, pero a su vez han causado inconvenientes en otros. Hoy en día se han vuelto muy populares las metodologías ágiles, la filosofía de este método se basa en el factor humano es decir darle mayor valor al individuo a la comunicación frecuente con el cliente y al progreso incremental al software con repeticiones muy cortas. Esta forma demuestra ser efectiva en diversos proyectos que contienen requisitos muy variantes y además cuando te encuentras obligado a reducir el tiempo de desarrollo detalladamente, pero conservando un alto nivel de calidad, estas metodologías están alterando la forma de crear programas creando una amplia discusión por sus partidarios y algunos que por desconfiados no las desean como elección en comparación de las metodologías estructuradas o formales.

La complejidad del desarrollo de sistemas de software es algo que siempre está presente de lo contrario todas las empresas y todos los clientes tendrían el software ideal sin errores. Sin embargo, un punto simple para controlar la dificultad inmanente en el software es tener un tipo de proceso en que guiarse, cuál será el más eficiente es lo que se describirá en el siguiente contenido.

MARCO TEÓRICO

1. METODOLOGÍA DE DESARROLLO

En un concepto amplio es “un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar nuevo software”. Normalmente consistirá en fases o etapas descompuestas en sub fases, módulos, etapas, pasos, etc. Esta descomposición ayuda a los desarrolladores en la elección de las técnicas a utilizar en cada estado del proyecto, facilitando la planificación, gestión, control y evaluación de los proyectos.” (Cataldi, 2003) Además hay que tener en cuenta que “la gestión de desarrollo de software es idiosincrático debido a la complejidad, conformidad, costes, flexibilidad e invisibilidad del propio software”(Ahimbisibwe, Cavana, & Daellenbach, 2015)

Algo que hay que tomar en cuenta es que “todos los tipos de desarrollo, excepto completamente secuencial, tienen iteraciones. Definimos los procesos en la iteración como "proceso de producción de valor" o, especialmente en agile "usuario proceso de la historia ". Cada requisito de usuario pequeño se forma como código en el proceso de producción de valor. La producción de valor puede definirse como una unidad de proceso que proporciona cierto valor para los usuarios. Hay dos tipos de valor proceso de producción: "completo" e "incompleto". El primero tiene un pequeño proceso de desarrollo secuencial completo excepto la definición / evolución del requisito, mientras que el último no. Además, las necesidades de los usuarios se definen una vez y evolucionó apropiadamente en el desarrollo híbrido con un proceso altamente iterativo.”(Jinzenji, Hoshino, Williams, & Takahashi, 2013)Lo que se está dando a conocer aquí es de vital importancia ya que se trata de mantener satisfecho en su totalidad a los clientes.

2. METODOLOGÍA ÁGIL

2.1. Antecedentes

En sus antecedentes de este método tenemos que “En febrero de 2001, tras una reunión celebrada en Utah EE. UU., nace el término “ágil” aplicado al desarrollo de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades. Tras esta reunión se creó The Agile Alliance, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía “ágil” “ (Ailin Orjuela Duarte, 2008)

2.2 Definición

En su definición tenemos según(López, 2016) “El término ágil surge como iniciativa de un conjunto de expertos en el área de desarrollo de software con el fin de optimizar el proceso de creación del mismo, el cual era caracterizado por ser rígido y con mucha documentación. El punto de partida fue el manifiesto ágil, el cual es un documento donde se detalla todo lo que involucra la filosofía “ágil”.

Dando un concepto concreto como lo establece (John Diaz, 2010)“En ingeniería de software, las metodologías ágiles utilizan los fundamentos del desarrollo iterativo evolutivo y del desarrollo adaptativo [6]. El primero implica que los requerimientos, los planes, las estimaciones y las soluciones evolucionan o son refinados a través de las iteraciones”.

"Esta metodología ágil está regida además por doce principios que ayudan a que el proceso de desarrollo se vuelva menos complejo y responda de manera oportuna a los cambios que surgen a lo largo del mismo, siempre contando con el punto de vista del cliente."(López, 2016)

En comparación del método ágil y el método tradicional es que "Los métodos ágiles se basan en Métodos de desarrollo de software adaptable, mientras que los modelos de tradicionales en un enfoque predictivo. Enfoques Cascada y Ágil tienen sus ventajas y desventajas si en comparación con las diferentes características del proyecto."(Singhto & Denwattana, 2016) Por esta razón el uso de estos métodos se está extendiendo pero según (Elgueta, 2016) "En la actualidad los métodos y metodologías Ágiles están siendo difundidos en la academia, sin embargo, en el contexto de la empresa tradicional se genera algunas dudas en el ámbito del desarrollo de software, porque rompe con los paradigmas existentes, es decir, la forma habitual de hacer las cosas."

3. METODOLOGÍA TRADICIONAL

En la otra del desarrollo tenemos al método tradicional que establece que "Al inicio el desarrollo de software era artesanal en su totalidad, la fuerte necesidad de mejorar el proceso y llevar los proyectos a la meta deseada, tuvieron que importarse la concepción y fundamentos de metodologías existentes en otras áreas y adaptarlas al desarrollo de software. Esta nueva etapa de adaptación contenía el desarrollo dividido en etapas de manera secuencial que de algo mejoraba la necesidad latente en el campo del software." (Roberth G. Figueroa, 2008)

Otro factor importante que hay que tener en cuenta de este método son "los altos costos al implementar un cambio y al no ofrecer una buena solución para proyectos donde el entorno es volátil." (Roberth G. Figueroa, 2008)

Además, algo más que acotar a este método son las pruebas. “Las pruebas de software tradicionales utilizan técnicas múltiples de pruebas técnicas, Combinada con dinámicas procesales y organizativas, para asegurar la calidad De una variedad de aspectos del software. Aquí vamos más allá de la estrecha Definición de pruebas de software para incluir conceptos como Verificación y Validación”(Shelly & Barta, 2010)

4. METODOLOGÍA XP (PROGRAMACIÓN EXTREMA)

4.1. Definición

Lo que se puede decir de este tipo de método que “es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los programadores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes.” (Ailin Orjuela Duarte, 2008)

También hay que resaltar que la metodología XP según (Joskowicz, 2008) “define cuatro variables para cualquier proyecto de software: costo, tiempo, calidad y alcance. Además, se especifica que, de estas cuatro variables, sólo tres de ellas podrán ser fijadas arbitrariamente por actores externos al grupo de desarrolladores (clientes y jefes de proyecto). El valor de la variable restante podrá ser establecido por el equipo de desarrollo, en función de los valores de las otras tres. Este mecanismo indica que, por ejemplo, si el cliente establece el alcance y la calidad, y el jefe de proyecto el precio, el grupo de desarrollo tendrá libertad para determinar el tiempo que

durará el proyecto. Este modelo es analizado por Kent Beck, en [9], donde propone las ventajas de un contrato con alcances opcionales.”

De acuerdo con (Dos Santos & Canedo, 2014), "A Extreme Programming, (programación extrema) emplea un enfoque orientado a como un paradigma de desarrollo preferido e implica un conjunto de reglas y prácticas constantes contexto de cuatro actividades metodológicas: planificación, diseño, codificación y pruebas ".

4.2. Prácticas

“La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. XP apuesta por un crecimiento lento del costo del cambio y con un comportamiento asintótico. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las prácticas.” (Letelier, 2006)

Este tipo de metodología se generaliza en doce principios básicos, de los cuales se conjuntan en 4 niveles entre las cuales se mencionan a continuación.

“Retroalimentación a Escala Fina, en esta fase se encuentran diversos principios como los de realización de pruebas, proceso de planificación, el cliente en el sitio y programación en parejas. Proceso Continuo en lugar de por lotes, permite la integración continua, refactorización (Evaluar el diseño del sistema a lo largo de todo el proyecto y codificar si es necesario) y entregas pequeñas. Entendimiento compartido, en esta categoría se definen criterios como el de crear un diseño fácil, las tarjetas CRC (Clase, Responsabilidad y PROCEEDINGS OF THE 2015 IEEE THIRTY FIFTH CENTRAL AMERICAN AND PANAMA CONVENTION (CONCAPAN XXXV) Colaboración) y la creación de la metáfora del sistema o historia completa. Bienestar del programador, se rige por la filosofía que un programador cansado, exhausto crea código de mala

calidad, por eso se recomienda que los desarrolladores tengan 40 horas de trabajo a la semana y muy pocas horas extras de trabajo."(Lopez, 2016)

5. METODOLOGÍA RUP

Entre sus antecedentes podemos decir que "Este modelo es desarrollado por I. Jacobson, G. Booch y J. Rumbaugh; Y Mantenido y Mejorado por Rational Software Corporation. RUP Es un enfoque iterativo para sistemas orientados a objetos. En un nivel alto, el proceso unificado está organizado Alrededor de dos conceptos: fases y flujos de trabajo."(Kumar & Bhatia, 2014)

En un concepto general se dice que "RUP es una metodología que tiene como objetivo ordenar y estructurar el desarrollo de software, en la cual se tienen un conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema Software (Amo, Martínez y Segovia, 2005). Inicialmente fue llamada UP (Unified Process) y luego cambió su nombre a RUP por el respaldo de Rational Software de IBM. Esta metodología fue lanzada en 1998 teniendo como sus creadores a Ivar Jacobson, Grady Booch y James Rumbaugh." (A., 2011).Esta metodología también "ha sido Como la metodología raíz para derivar la propuesta Uno debido a su capacidad para soportar enfoque incremental.(John, 2010)

5.1. El ciclo de vida del RUP

RUP también tiene su ciclo de vida donde "se repite a lo largo de una serie de ciclos (que constituyen la vida de un sistema desde su nacimiento hasta su muerte. Cada ciclo concluye con una versión del producto para los clientes, el ciclo de vida de RUP está comprendido por varios ciclos. Las versiones y ciclos le añaden funcionalidad al sistema hasta el punto donde termine su ciclo de vida con la muerte o cumplimiento total del objetivo para el cual fue diseñado el software. Los elementos que conforman el proceso interno de un ciclo. Los cuales comprenden

las fases y sus respectivas iteraciones, a su vez cada ciclo concluye con una versión del sistema software.” (A., 2011)

5.2. Características principales de RUP

Las características principales que vamos a mencionar a continuación fueron sacadas de (Expósito, 2008) de las cuales son:

“Centrado en los modelos: Los diagramas son un vehículo de comunicación más expresivo que las descripciones en lenguaje natural. Se trata de minimizar el uso de descripciones y especificaciones textuales del sistema.

Guiado por los Casos de Uso: Los Casos de Uso son el instrumento para validar la arquitectura del software y extraer los casos de prueba. [10]

Centrado en la arquitectura: Los modelos son proyecciones del análisis y el diseño constituye la arquitectura del producto a desarrollar.

Iterativo e incremental: Durante todo el proceso de desarrollo se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo.

5.3 Beneficios que aporta RUP

Permite desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, el rendimiento, la reutilización, la seguridad y el mantenimiento del software mediante una gestión sistemática de riesgos.

Da lugar a la producción de software que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos, con una agenda y costo predecible. Enriquece la productividad en equipo y proporciona prácticas óptimas de software a todos sus miembros.” (Expósito, 2008)

DESARROLLO

METODOLOGÍA TRADICIONAL

A principios el desarrollo de un software era artesanal, debido a la intensa necesidad de perfeccionar los procesos de un proyecto y llevar al objetivo deseado, es por esta razón que tuvieron que darle importancia a la concepción y fundamentos de metodologías reales en diferentes campos y acordarlas a un software de desarrollo. Esta etapa desconocida de adaptación abarcaba el desarrollo en el que estaba divida en etapas secuenciales la cual mejoraba la necesidad oculta en el campo del software.

Una de las metodologías más tradicionales e importantes tenemos los llamado RUP y MSF, que consiste en llevar una documentación absoluta de todo el proyecto y su atención se fija en desempeñar con un plan de proyecto, y todo esto se da en la fase inicial del desarrollo del proyecto.

Cabe resaltar otras particularidades primordiales, por ejemplo, dentro de este punto de vista tenemos los altos costos al momento de hacer un cambio y no brindar un resultado específico con un entorno ágil para ciertos proyectos.

Las metodologías que se enfocan en la documentación, planificación y procesos como se nombró anteriormente son las llamadas tradicionales o formales y dentro de esta el método RUP, del cual es uno de los métodos más utilizados.

Rational Unified Process “RUP”

Desarrollado por Rational Software e integrado por la suite Rational de herramientas, a su vez es adaptado y extendido para soportar los objetivos de la organización de los cuales vayan a utilizar esta metodología. Además, es centrado en la arquitectura y orientado por casos de uso, es por ello, que usa UML como lenguaje de notación, que en español significa Proceso racional

unificado, donde provee un contorno sumiso con la finalidad de determinar responsabilidades y tareas en una organización de desarrollo. Uno de sus principales objetivos es consolidar la producción de software de primera calidad donde se requiere satisfacer las necesidades de los usuarios finales y además se respeta el importe inicial y cronograma.

Fases

La metodología RUP consta de cuatro fases del ciclo de vida:

- Concepción
- Elaboración
- Construcción
- Transición

Existen ventajas y desventajas para esta metodología, entre ellas tenemos las siguientes:

Ventajas

- Búsqueda profunda en cada una de las fases.
- En proyectos de innovación funcionan a la perfección.
- Se puede decir que es simple, debido a que sigue pasos intuitivos y precisos al momento de desarrollar el software.
- Analizar los cambios permitiendo modificar los objetivos.

Desventajas

- El cliente deberá tener la capacidad suficiente para describir y entender a un alto nivel de detalle referente a un alcance del proyecto.
- Para unos proyectos tendrá que tener excesiva flexibilidad.
- Cuando existe la oportunidad de evaluar riesgos se torna compleja.
- Al cliente se le brinda una situación que puede ser muy molestosa para él.

METODOLOGÍAS ÁGILES.

Después de varios dictámenes tanto a favor como en contra de las metodologías tradicionales nace un nuevo enfoque llamado, métodos ágiles, que conlleva a la respuesta a los problemas detallados inicialmente y se sustenta en dos puntos claves, el primero es retardar las decisiones y la otra es tener una planificación que sea adaptable; obteniendo como efecto un sistema realizado con un alto nivel.

A partir de esta nueva teoría da como resultado un Manifiesto Ágil cuyos principales objetivos son:

- No es primordial el seguimiento estricto de un plan si no las varias correcciones para obtener un cambio positivo.
- Es de suma importancia que al crear un producto de software funcione de manera correcta a que se escriba una documentación exhaustiva.
- Son más importantes los individuos y las interacciones a que las herramientas y los procesos empleados.
- Siempre debe sobresalir la colaboración con el cliente con respecto a la negociación de contratos.

Los métodos agiles más utilizados son: XP “eXtreme Programming”, Cristal Methods , AUP y Scrum.

La capacidad de respuesta a un cambio es el entorno básico para estas metodologías que un seguimiento estricto de un plan, debido a que muchos clientes se les brinda una utilidad bastante competitiva y porque estar dispuesto para el cambio conlleva a reducción de costo.

Retrasos de decisiones y Planificación Adaptable

Estas características son primordiales para la metodología ágil, porque cuando hablamos de retrasar decisiones tan como sea necesario y de forma responsable el beneficiado de esto será tanto el cliente como la empresa, lo cual da lugar a una satisfacción de nivel alto al cliente y por lógica conllevará el éxito del producto, a continuación de darán a conocer las ventajas que destacan para lograr el retraso en dichas decisiones:

- Reducir el coste del cambio.
- Reducir el número de decisiones de alta inversión que se llegue a tomar.
- Reducir el número de cambios en el proyecto que se esté ejecutando.

La planificación adaptable nos permite estar listos para el cambio ya que se lo ha introducido en el proceso de desarrollo, además al realizar una planificación adaptable genera la toma de decisiones a lo largo del proyecto, donde se irá transformando el proyecto en una colección de proyectos aún más pequeños.

Cuando se tiene una planificación a corto plazo se nos facilitara la obtención de un software para nuestros clientes y con ello ir aprendiendo con lo que respecto a feedback, para que una planificación sea más afectiva, ante riesgos ya sea que aceleren o retrasen el producto.

En esta parte se hablará acerca de XP que es el más admitido dentro del desarrollo de SW

Extreme Programming "XP"

Creada por Kent Beck y es la más manifestada por parte de los procesos ágiles, comúnmente esta programación extrema es muy diferente al resto de las metodologías tradicionales con respecto al énfasis de la adaptación que en la previsibilidad.

Las personas que abogan por XP suponen que los cambios de requisitos con respecto a la marcha tienen un aspecto natural, inevitable e inclusive deseable del desarrollo de proyectos. Creen tener la capacidad de poder adaptarse a los cambios de requisitos en cualquier momento

de la vida del proyecto, es mejor ser realista que intentar limitar todos los requisitos al principio del proyecto e invertir esfuerzos posteriormente cuando sea controlado los cambios en los requisitos.

Propiedades esenciales del método XP

Interactuar junto con un grupo de programación, adicionando al cliente o usuario. Es aconsejable tener un representante del cliente para que trabaje junto al equipo de desarrollo.

Programar en pareja es recomendable realizar tareas de desarrollo en grupo de dos personas en un mismo puesto, debido a que el código será revisado y discutido mientras se escribe.

Realizar pruebas continuas, casi siempre repetidas y automatizadas, adjuntando pruebas de regresión. Previamente se recomienda la prueba de la codificación, luego se da paso a escribir el código.

Antes de modificar alguna función se tiene que solucionar todos los inconvenientes donde se estén presentando, realizando entregas frecuentes.

La mejor forma que las cosas salgan satisfactorias es requerir con un código simple ya que la programación extrema nos dice que en más sobrio hacer algo sencillo y tener trabajo extra para innovar si se presenta la oportunidad, que elaborar algo difícil y quizás nunca se lo utilice.

Fomentar de manera repetitiva, es decir tener progresos pequeños, unas tras otras.

Re factorizar código, en otras palabras, reescribir ciertas partes del código con finalidad de elevar su legibilidad y mantenibilidad, siempre y cuando no se modifique su comportamiento. Al momento de efectuar las pruebas se debe garantizar que en la refactorización no se cometido o introducido ningún fallo al respecto.

Cumplir con el método de Propiedad de código compartida, en pocas palabras, que todo el personal tenga la oportunidad de corregir y extender cualquier parte del proyecto, por lo

contrario de dividir el trabajo del progreso de cada nivel en los diferentes equipos de trabajo. Las pruebas periódicas de regresión nos van a garantizar que los posibles deslices serán detectados.

La característica imprescindible es tener una buena comunicación y a la vez una simplicidad profunda, porque entre más comunicación se tenga el resultado de poder reconocer lo qué se debe y qué no se debe hacer se hará más factible, además no tendrá que comunicar sobre este, lo que nos conlleva a una comunicación más completa, en particular si se da la ocasión de reducir el equipo de programadores.

Ventajas

- Es adecuado para contextos volátiles
- Un cambio significa reducir su coste.
- Una vital característica para un negocio es brindar mayor transparencia posible para nuestros clientes, así tienen noción de las fechas de entrega de funcionalidades.
- Definen en cada iteración los objetivos de la siguiente.
- Los usuarios útiles tienden a tener realimentación.
- La presión se encuentra a lo largo de todo el proyecto y no al final.

Desventajas

- No se podrá definir el alcance del proyecto con el usuario.
- Demasiada complejidad y libertad al cliente
- Imposibilidad de previsión global.

Para poder descartar esta desventaja se propone concretar un alcance de un alto nivel basado en la experiencia.

CONCLUSIONES

Las metodologías de desarrollo son básicamente una serie de prácticas o métodos de trabajo, que se han venido analizando hace mucho tiempo atrás, para favorecer el desarrollo del software, siempre y cuando se logre la optimización de los procesos, es decir, que sean muchos más agiles, que tengan resultados más pronto y por supuesto de calidad, y que los beneficiados sean los usuarios y quien este elaborando dicho proyecto. Estas metodologías no son siempre prefectas, ya que cuentan con una serie de ventajas y desventajas, unas sobre otras, muchas son muy usadas y otras no; dependiendo del tipo de proyecto, pueden resultar conveniente utilizar según a su criterio, hay muchos factores que involucran al momento de elegir la metodología correcta, ya que no existe una táctica universal para el desarrollo de software en la que se pueda aplicar y con la que podamos hacer todos los proyectos, es por eso que se recomienda analizar con base al proyecto que tenemos en frente con la finalidad de elegir la norma más adecuada.

Sin embargo, con el avance de los años se ha encontrado diferentes metodologías, pero entre las más usadas tenemos las estructuradas o también llamadas tradicionales, este modelo se basa en organizar los procesos de arriba hacia abajo, dicho de otro modo, antes de que inicie uno debe haber terminado el anterior, no obstante se presentan desventajas para este método como la susceptibilidad ante errores y fallos, tiempos de ejecución demasiado largos y por si no fuera poco es inflexible y menos adaptable a cambios cuando el usuario amerite. Por otro lado existe la Metodología Ágil, que hoy en día se han vuelto muy utilizadas y populares debido al controla de errores y a la sencillez, tanto en su práctica como en su manejo, cabe destacar que una buena comunicación es el rol importante para este caso, debido a la interacción que se da entre usuario y programador o jefe o como el papel que cumpla dentro de una organización, a su vez hace énfasis a la adaptabilidad de nuevos requisitos que tienden a tener los proyectos, también son

considerados los cambios de los requerimientos, cuando el cliente lo vea necesario, puesto que son algo completamente natural dentro de este modelo, asimismo no hay que dejar pasar por alto la realimentación, esta es indispensable, su objetivo primordial es adaptarse a lo que el cliente desee, es factible que cada parte realimente a la otra, para evitar cualquier inconveniente, dándole la posibilidad al sistema de crecer y de evitar errores. Una de las practicas más comunes que se deben seguir para una mejor funcionalidad y que lleven una secuencia ordenada en esta modalidad son: las pruebas, metáforas, programación en parejas, diseños simples, refactorización, estándares de programación, entre otros, entonces podemos concluir que la metodología más empleada por parte de los programadores o persona que esté a cargo del proyecto es la Ágil, debido a sus diferentes cualidades que nos brinda dicho modelo, haciendo de esta un mejor desarrollo de Software.

BIBLIOGRAFÍA

- A., O. A. (2011). Cuatro enfoques metodológicos para el desarrollo de Software RUP – MSF – XP - SCRUM. *Inventum*, 2. Obtenido de http://biblioteca.uniminuto.edu/ojs/index.php/Inventum/article/view/9/9
- Agustin Yagüe, J. G. (2009). *Las pruebas en metodologías ágiles y convencionales: papeles.* Madrid-España: Universidad Politecnica de Madrid (UPM) .
- Ailin Orjuela Duarte, M. (2008). Las Metodologías de Desarrollo Ágil como una Oportunidad. *Avances en Sistemas e Informaica*, 162. Obtenido de http://www.redalyc.org/pdf/1331/133115027022.pdf
- Cataldi, Z. (28 de Noviembre de 2003). *SEDICI.* Obtenido de http://sedici.unlp.edu.ar/handle/10915/4055
- Expósito, I. E. (2008). Metodologías de desarrollo de software. ¿Cuál es el camino? *Revista de Arquitectura e Ingeniería*, 2. Obtenido de http://www.redalyc.org/pdf/1939/193915935003.pdf
- Hernán., S. M. (2004). Diseño de una Metodología Ágil de. BUENOS AIRES, Argentina: Universidad de Buenos Aires. .
- John Diaz, J. O. (2010). *ConstruColectiva: Guía metodológica para la gestión de.* Bogota D.C: Universidad Javeriana. Obtenido de https://www.researchgate.net/profile/Consuelo_Franky/publication/267808702_ConstruColectiva_Guia_metodologica_para_la_gestion_de_proyectos_de_software_basados_en_metodologias_agiles_utilizando_ambientes_de_desarrollo_colaborativo/links/5474cae70cf2778985a

- Joskowicz, I. J. (10 de 02 de 2008). Reglas y Prácticas en Extreme Programming. España: Universidad.
- Letelier, P. (Junio de 2006). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. Buenos Aires, Argentina. Obtenido de http://www.cyta.com.ar/ta0502/b_v5n2a1.htm
- Roberth G. Figueroa, C. J. (2008). *Metodologias Tradicionales VS Metodologias Agiles.* Loja: Universidad Tecnica Particular de Loja.
- Valdéz, J. L. (8 de Mayo de 2014). IMPLEMENTACIÓN DEL MODELO INTEGRAL COLABORATIVO (MDSIC) COMO FUENTE DE INNOVACIÓN PARA EL DESARROLLO ÁGIL DE SOFTWARE EN LAS EMPRESAS DE LA ZONA CENTRO - OCCIDENTE EN MÉXICO. Mexico. Obtenido de http://www.eumed.net/tesis-doctorales/2014/jlcv/ficha.htm

- Ahimbisibwe, A., Cavana, R. Y., & Daellenbach, U. (2015). A contingency fit model of critical success factors for software development projects: A comparison of agile and traditional plan-based methodologies. *Journal of Enterprise Information Management*, *28*(1). https://doi.org/10.1108/JEIM-08-2013-0060
- Dos Santos, V. S. A., & Canedo, E. D. (2014). Agile methodology in the software development: Case study: Electoral justice of Brazil | Metodologia ágil no desenvolvimento de software: Estudo de caso: Justiça eleitoral do Brasil. In *Iberian Conference on Information Systems and Technologies, CISTI*. https://doi.org/10.1109/CISTI.2014.6876981

- Elgueta, J. C. (2016). Methods agile and methodology A+S in the teaching of software engineering | Métodos Ágiles y Metodología A+S en la enseñanza de Ingeniería de software. In *2016 IEEE International Conference on Automatica, ICA-ACCA 2016*. https://doi.org/10.1109/ICA-ACCA.2016.7778512
- Jinzenji, K., Hoshino, T., Williams, L., & Takahashi, K. (2013). An experience report for software quality evaluation in highly iterative development methodology using traditional metrics. In *2013 IEEE 24th International Symposium on Software Reliability Engineering, ISSRE 2013*. https://doi.org/10.1109/ISSRE.2013.6698884
- John, S. (2010). Leveraging traditional software engineering tools to ontology engineering under a new methodology. In *2010 5th International Conference on Future Information Technology, FutureTech 2010 - Proceedings*. https://doi.org/10.1109/FUTURETECH.2010.5482657
- Kumar, G., & Bhatia, P. K. (2014). Comparative analysis of software engineering models from traditional to modern methodologies. In *International Conference on Advanced Computing and Communication Technologies, ACCT*. https://doi.org/10.1109/ACCT.2014.73
- Lopez, R. E. (2016). Agile software development applied to the management of business projects. In *Proceedings of the 2015 IEEE 35th Central American and Panama Convention, CONCAPAN 2015*. https://doi.org/10.1109/CONCAPAN.2015.7428451
- Shelly, C. C., & Barta, M. (2010). Application of traditional software testing methodologies to web accessibility. *Proceedings of the 2010 International Cross*

Disciplinary Conference on Web Accessibility (W4A) - W4A '10, 1. https://doi.org/10.1145/1805986.1806002

- Singhto, W., & Denwattana, N. (2016). An experience in blending the traditional and Agile methodologies to assist in a small software development project. In *2016 13th International Joint Conference on Computer Science and Software Engineering, JCSSE 2016*. https://doi.org/10.1109/JCSSE.2016.7748914