INVESTIGACIÓN

DISEÑO DE UN MÉTODO ÁGIL DE DESARROLLO DE SOFTWARE BASADO EN XP, SCRUM, OPENUP Y VALIDADO CON LA HERRAMIENTA DE ANALISIS 4-DAT

AGILE METHOD DESIGN FOR SOFTWARE DEVELOPMENT ON XP, SCRUM, OPENUP AND ANALISYS TOOL 4-DAT VALIDATED

Ing. Jessica Patricia Gamboa carrascal[a], MSc. Alveiro Rosado Gómez[b]

[b] Universidad Francisco de Paula Santander Ocaña, vía Acolsure, Sede el Algodonal, Ocaña, Colombia, jpgamboac@ufpso.edu.co
[c] Universidad Francisco de Paula Santander Ocaña, Grupo de Investigación en Tecnología y Desarrollo en Ingeniería GITYD, Via Acolsure, Sede el Algodonal, Ocaña, Colombia, aarosadog@ufpso.edu.co

Fecha de recepción: 13-06-2015 **Fecha de aprobación:** 10-11-2015

Resumen: Esta investigación propone el diseño de una alternativa ágil para el desarrollo de software que integrara las mejores prácticas de XP, Scrum y OpenUP. Se comenzó definiendo y conociendo los roles, ciclo de vida y artefactos de los tres métodos, desde esos tres criterios se hizo un mapeo entre sus características, para crear a partir de ellas la nueva propuesta. Para determinar el grado de cumplimiento del manifiesto ágil, el método fue evaluado con la herramienta de análisis y comparación 4-DAT, la cual arrojo como resultado que el nuevo método cumple con todos los requisitos para considerarse ágil y por lo tanto estructurar los elementos necesarios para el desarrollo de software.

Palabras clave: Desarrollo de software, herramienta de análisis dimensional, método ágil, Scrum, OpenUP, XP.

Abstract: This research proposed the design of a new agile software development method to integrate of XP, Scrum and OpenUP best practices. This project was developed began defining and knowing the roles , life cycle and artifacts of the three methods from this comparison was made and a mapping between their characteristics which were then integrated into the proposed new method, which It was assessed with the analysis and comparison tool 4- DAT , which is responsible for checking all requirements to be considered an agile software development method.

Keywords: Software development, dimensional analysis tool, agile method, Scrum, Open UP, XP.

1. INTRODUCCIÓN

A través del tiempo, el desarrollo de software ha ido escalando y mejorando, existiendo una gran variedad de procesos para su solución, esto en parte se logró con la aplicación de metodologías de desarrollo, las cuales no son más que un conjunto de procedimientos, técnicas, herramientas y documentos que tienen como fin ayudar a los desarrolladores de Software en la administración de sus proyectos (Tinoco, Rosales Lopez, & Salas Bacalla, 2010).

Existen ciclos de vida tradicionales y métodos agiles, los primeros requieren elementos más completos y complejos que los que especifican los segundos (Bennett, McRobb, & Farmer, 2006). Los métodos agiles se basan en el desarrollo "iterativo e incremental", lo cual produce resultados en corto lapso de tiempo, son flexibles a cambios y minimizan la documentación (Kendall & Kendall, 2005).

La variedad de métodos ágiles que existen produjo que los desarrolladores se vean enfrentados a la decisión de elegir cuál de ellos utilizar, eligiendo al que garantice el éxito del proceso con un menor número de errores posibles teniendo como criterios el tamaño y la complejidad del proyecto (Maurer & Melnik, 2006).

Los métodos agiles tienen mucha acogida en empresas que desarrollan software, es así que un estudio realizado en el 2013 (VersionOne, 2014), a un grupo de empresas de este sector, encontró que el 88% de las organizaciones están aplicando estos métodos, mostrando un crecimiento del 4% con respecto al 2012, esto indica que los métodos ágiles tienen aceptación y crecimiento en pequeñas y grandes empresas; además el estudio permite determinar que el 19% de las empresas que empezaron a usarlos lo ha seguido aplicando durante los últimos 5 años, la razones por las cuales los siguen usando es que son estables, completas y una buena opción para gestionar proyectos.

El uso de métodos agiles para el desarrollo de proyectos de software proporciona una serie de ventajas, entre las que se encuentran: disminución de documentación, mejor relación con el cliente; ya que está involucrado totalmente en el proyecto, promueve el trabajo en equipo; con sus reuniones diarias, mejora la calidad del producto; dado que no se pasa a la siguiente fase hasta que se hayan corregido todos los errores (Highsmith, 2002) (Kendall & Kendall, 2005).

En este proyecto se usaron tres métodos agiles como referencia para crear el nuevo método, se empieza por Extreme Programming (Xp), el cual busca reducir el uso exhaustivo de documentación por lo cual orienta su trabajo directamente al objetivo, y se basa en las relaciones cara a cara con el cliente y la capacidad de reaccionar rápidamente ante los cambios (Kent, 1999).

Otro método que se utilizó fue Scrum, el cual es un marco de desarrollo que consta de equipos multifuncionales y que trabajan de manera iterativa. Entre las características que se encuentran en los grupos de trabajo que aplican Scrum se encuentran consientes del cambio que se puede producir en

cualquier momento del desarrollo y que se genera por la cultura de la organización; por esta razón los equipos de desarrollo son pequeños, auto-dirigidos y auto-organizados, las fases de desarrollo son permanentemente controladas, la difusión y cumplimiento de las tareas se lleva mediante reuniones diarias (Mendes Calo, Estevez, & Fillottrani, 2009). El desarrollo de su ciclo de trabajo se hace mediante sprints, que son iteraciones cuya duración no excede las cuatro semanas, al comienzo de los sprints los grupos de trabajo seleccionan una lista con los requisitos del cliente y le dan una prioridad que se basa en lo que cree puede ofrecer al final del sprint un elemento tangible y completamente terminado (Deemer, Benelfield, Larman, & Vodde, 2012).

El tercer método que se utilizo fue OpenUP, el cual es un proceso unificado que aplica enfoques iterativos e incrementales dentro de un ciclo de vida estructurado, se centra en la naturaleza colaborativa de desarrollo de software (Kroll & MacIsaac, 2006). Un artefacto en OpenUP se considera a todo aquello que una tarea necesita para realizar su función, o bien la produce o modifica. Los distintos roles existentes son los encargados de crear y actualizar los artefactos (Eclipse Foundation, 2012).

Se decidió trabajar sobre estos métodos ya que tienen mayor acogida (VersionOne, 2014), y cumplen ampliamente con características como vista del sistema como algo cambiante, tener en cuenta la colaboración entre los miembros del equipo, simplicidad, excelencia técnica, resultados y adaptabilidad (Highsmith, 2002).

Si bien VersionOne (2014) y Highsmith (2002), determinan a XP y Scrum, como las comúnmente aplicadas, se decidió también incluir OpenUP ya que posee mayor número de procesos y artefactos que un método ágil y menos que un método tradicional (Pillat, Oliveira, & Fonseca, 2012), lo cual le agrega un valor adicional a la nueva propuesta ya que uno de los inconvenientes que poseen los métodos agiles es que no aplican algunos artefactos, relevantes para el diseño de la aplicación (Hadar & Sherman, 2012). Además emplea plantillas para el desarrollo lo cual hace que se mantenga mayor organización en el proyecto (Kroll & MacIsaac, 2006).

2. METODOLOGÍA

Para la construcción de la nueva propuesta ágil se partió de las características que estos tres métodos tienen en común desde el punto de vista del ciclo de vida, los roles y los artefactos (Nazareno, Leone, & Gonnet, 2013). Primero se estableció los elementos que tienen en común los tres métodos agiles, indicando los aspectos permanentes en cada uno de ellos y luego estableciendo un mapeo entre cada aspecto con el elemento del método ágil que lo desarrolla. A continuación se describe el resultado de las similitudes que existen entre los tres métodos.

2.1 Ciclo de vida Xp, Scrum Y OpenUP:

Cada método ágil utiliza un ciclo de vida para el desarrollo del proyecto esto lo hace

para definir etapas específicas y controlar la organización y manejo del proyecto.

En la tabla 1, se muestra la relación que existe entre el ciclo de vida que utiliza cada método y la fase correspondiente que lo abarca.

Tabla 1. Relación entre ciclos de vida

CICLO DE VIDA			
RELACION EXISTENTE	**XP**	**SCRUM**	**OPENUP**
Se describen objetivos, se asigna prioridad, se realiza la arquitectura.	Fase de exploración Fase de planificación de la entrega (reléase)	Fase de planeamiento Fase de montaje	Fase de inicio Fase de elaboración
Desarrollo del producto y entregable de prueba.	Fase de iteraciones Fase de producción	Fase de desarrollo	Fase de construcción
Liberación del producto, realización de documentación	Fase muerte del proyecto.	Fase de liberación	Fase de transición

Fuente. Autores del proyecto

Al compararlas se puede observar que las dos primeras etapas de cada uno de ellos realizan lo mismo; lo cual es definir objetivos, requisitos o casos de uso, asignar prioridad, se establece el tiempo de desarrollo de cada uno de ellos, y se hace un prototipo de la arquitectura, en la fase siguiente se tiene el desarrollo del sistema y un entregable que se pone a prueba mientras Scrum y OpenUP lo realiza en una sola fase, XP lo separa en dos que son iteraciones y producción.

Después de haber definido objetivos y demás actividades de la primera fase Scrum y OpenUP liberan el producto, capacita a los usuarios finales y realiza la debida documentación. En cuanto XP, antes de realizar esto asigna al cliente tareas de soporte y si es necesario se hace cambio de estructura para luego proceder con la fase final.

2.2 Roles XP, Scrum Y OpenUP:

Un equipo de desarrollo debe tener diferentes expertos para lograr un producto con mayor calidad; dado que al presentarse cualquier inconveniente sea solucionado inmediatamente por quien esté a cargo (Maurer & Melnik, 2006).

A continuación la tabla 2 muestra la relación existente entre cada uno de los roles de cada método ágil. El mapeo se establece entre la relación que existe y los roles que los cumplen o complementan en cada método.

Tabla 2. Relación entre roles

ROLES			
RELACION EXISTENTE	**XP**	**SCRUM**	**OPENUP**
Programación y desarrollo del producto	Desarrollador	Equipo de desarrollo	Arquitecto y desarrollador.
Requisitos del sistema	Cliente	Product owner	Cliente
Guía de proceso	Coach	Scrum master	Gestor del proyecto
Seguimiento del desarrollo de trabajo.	Tracker	Product owner	Ingeniero de implementación y arquitecto
Vinculo cliente-equipo	Big boss Tester	Scrum master	Analista
Pruebas de funcionamiento	Tester	Equipo de desarrollo	Tester Gestor de despliegue

Fuente. Autores del proyecto

Al relacionarlos se puede observar que OpenUP maneja un número mayor de roles a diferencia de XP y Scrum que son menos explícitos. Esto no quiere decir que algún método tenga roles mal gestionados cada uno tiene un fin específico (Elvesæter & Benguria, 2013) y puede ser comparado con otro de cualquier método diferente pero que cumpla la misma función, con esto se hace referencia a lo siguiente:

En cuanto a la programación y desarrollo del producto XP tiene al desarrollador, Scrum maneja estos procesos en el equipo de desarrollo, el cual está conformado por un grupo de profesionales, mientras que en OpenUP los encargados de manejar esto son el desarrollador y el arquitecto.

Para establecer los requisitos del sistema los tres métodos tienen el rol cliente, en XP el cliente es el encargado de establecer las historias de usuario y asignarles una prioridad, en Scrum se denomina el product owner que es el encargado de gestionar la product backlog y en OpenUP el cliente expone sus necesidades y el equipo se encarga de sacar los requisitos.

Cada uno de los métodos tiene un encargado de exponer los valores, prácticas y reglas y hacer que se cumplan en el caso de XP el responsable es el coach, en Scrum está el scrum master y en OPenUP este papel es desempeñado por el gestor del proyecto.

Cuando se refiere al seguimiento del proceso estos métodos tienen un rol asignado en XP el tracker es el encargado de hacer seguimiento a cada iteración, Scrum el product owner realiza esto y en OpenUP quien realiza esto es el ingeniero de implementación y el arquitecto.

Debe existir un vínculo entre el cliente y el equipo de desarrollo, este rol se encarga de coordinar y lograr que lo que pide el cliente sea cumplido, el encargado de esto en XP es el big boss y el tester quien también realiza otras funciones, en Scrum es el scrum master y en OpenUP es el analista.

Al proyecto que se desarrolla deben realizarle una serie de pruebas de funcionamiento antes de ser entregado para esto XP, dispone del tester que las ejecuta regularmente, Scrum lo hace el equipo de desarrollo y en OpenUP los encargados son el tester y el gestor de despliegue.

OpenUP es mucho más organizado y va más allá de XP y Scrum en cuanto a definir roles para la documentación y capacitación de cómo debe usarse el producto final, para esto cuenta con un escritor técnico, un desarrollador del curso y un entrenador (Khelladi, Bendraou, Baarir, Laurent, & Gervais, 2015).

2.3 Artefactos XP, Scrum Y OpenUP:

Un artefacto es todo aquello que se necesita para que una tarea sea realizada (Anaya, 2006), la tabla 3, muestra de que manera maneja cada método sus artefactos y de cuál es la relación que existe entre ellos.

Al compararlos se puede observar que cada uno de ellos utiliza artefactos totalmente diferentes pero igualmente funcionales mientras XP utiliza historias de usuario, Scrum maneja una lista del producto y OpenUP casos de uso y otra serie de

artefactos que son necesarios para que una tarea realice su función.

Los métodos agiles poseen características esenciales que los hace particulares, pero al analizarlos como se hizo con la comparación anterior, se puede notar que cada uno de ellos tiene un proceso y unas fases para planificar el desarrollo del proyecto que aunque no son iguales en cantidad o no poseen el mismo nombre tienen como finalidad establecer un patrón para el desarrollo del proyecto.

Tabla 3. Relación entre artefactos

ARTEFACTOS			
RELACION EXISTENTE	**XP**	**SCRUM**	**OPENUP**
Explicación de Lo que el producto o sistema debe realizar	Historias de usuario	Producto backlog	Plan del proyecto Proceso definido del proyecto Libro de arquitectura Diseño Visión Caso de uso Modelo de caso de uso Requisitos del sistema
Culminación y cumplimiento de requisitos.	Pruebas de aceptación	Sprint backlog	Prueba desarrollador Plan de despliegue

Fuente. Autores del proyecto

Otra característica es que para el desarrollo de los requerimientos cada método usa diferente tipo de escenarios, pero en cada uno de ellos se especifica claramente lo que se va a realizar y lo que desea el usuario.

Por otra parte cada método cuenta con un equipo de trabajo y aunque en algunos hay mayor número de integrantes, que en otros en cualquier caso siempre habrá alguien que desempeñe un rol específico en el desarrollo del proyecto.

De la misma manera en que se logró encontrar las similitudes entre los tres métodos se puede decir que en cada una de sus etapas existen ciertas diferencias que se expresan a continuación:

En cuanto a roles XP tiene un consultor que es un miembro externo al equipo con conocimientos en un tema específico sobre el cual puedan surgir problemas en algún momento. Scrum posee un management que realiza tareas de revisión y seguimiento del sprint backlog y por ultimo OpenUP que tiene un participante para cada tarea a realizar como ingeniero de procesos, escritor técnico, especialista en herramientas entre otros, por lo cual se diferencia totalmente de los otros métodos.

Al hablar de artefactos XP, desarrolla historias de usuario, usa tareas de ingenierías, las cuales son plantillas que se necesitan para hacerle seguimiento al desarrollo mientras que Scrum, a diferencia de ellos dos usa un artefacto llamado burndown chart que es una gráfica que muestra el progreso en el desarrollo de los requisitos. Por último OpenUP es más completo en cuanto a artefactos y posee documentación del usuario, del producto, un plan de backup, visión, comunicaciones de lanzamiento y demás.

3. RESULTADOS

3.1 Método propuesto: MAF (Métodos agiles fusionados)

Este nuevo método propuesto (En adelante MAF), es un método de desarrollo ágil, basado en una serie de prácticas y valores extraídos de XP, Scrum y OpenUP. Tiene como objetivo mejorar la productividad y calidad en los proyectos realizados por los desarrolladores de software y así mejorar su desempeño en el uso de técnicas agiles (Sharp, Biddle, Gray, Miller, & Patton, 2006).

MAF, está constituido por tres métodos agiles; tiene un ciclo de vida iterativo e incremental (Albaladejo, 2010), que hacen que el proyecto sea mejorado sucesivamente, MAF busca que los grupos de gestión de software manejen marcos de desarrollo ágil, tengan buena actitud frente a cambios repentinos del sistema, trabajen en equipo y manejen varios procesos en cada etapa de desarrollo.

Este método fue estructurado a partir de aspectos fundamentales como roles, artefactos, ciclo de vida y administración.

A continuación se expone los elementos que propone MAF, como método de desarrollo.

Ciclo de vida MAF:

Todo proyecto de desarrollo ágil de software debe tener un ciclo de vida que especifique los pasos y fases que se deben seguir para mantener un orden, XP, Scrum y OpenUP usan fases que tienen mucho en común, cada una a su manera se divide en diferentes partes pero en síntesis realizan las mismas actividades, a partir de eso MAF utiliza las siguientes etapas para su desarrollo:

La primera fase es la de planeación en ella se establecen los requisitos del usuario, el cliente expone todo lo que quiere en su producto y se crea el product backlog, el cliente es el encargado de dar prioridad a las historias de usuario, al mismo tiempo se estiman costos. Se examinan las posibilidades de la arquitectura del sistema realizando un prototipo. Se le asigna a cada miembro del equipo la tarea que debe realizar en la realización de cada historia de usuario. La segunda fase es la de desarrollo En esta fase en la primera entrega se incluyen varias iteraciones, la primera iteración crea la arquitectura del sistema, para esto se deben buscar las historias de usuario que cumplan con esos requisitos, el encargado de esto será el cliente. En esta fase se hacen encuentros de planeamiento y encuentros diarios para analizar el avance del proyecto. Por ultimo esta la fase de culminación, en donde se hace la liberación del producto y la realización de la documentación. Además de las anteriores MAF, cuenta con sub fases de pruebas y capacitación a los usuarios finales y de los encargados del mantenimiento del sistema.

Roles MAF:

Los roles definidos para MAF, corresponden a la integración de los establecidos en la tabla 2, pero con un nombre genérico que represente el alcance de su participación dentro del método.

Se tiene un cliente que es el encargado de establecer los requisitos y objetivos del sistema, un desarrollador que es el encargado de la programación del producto, el facilitador es quien da al equipo de trabajo una guía de como cumplir con los requisitos propuestos por MAF, existe un delegado que es el vínculo entre el cliente, desarrolladores y el equipo en general. Por último está el tester, que es el encargado de realizar pruebas de funcionamiento constantemente, con el fin de establecer el avance del desarrollo.

Artefactos MAF:

Los artefactos son los elementos primordiales para documentar el proceso de desarrollo (Anaya, 2006). En ellos se almacena las decisiones técnicas, los diseños y la especificación de funcionalidades, esto permite que eventualmente hacer correcciones, estimar la usabilidad, el mantenimiento futuro y las ampliaciones del sistema (Pressman, 2006).

Se pueden destacar tres elementos importantes al momento de usar artefactos en un método ágil:

- Explicación de lo que el producto o sistema debe realizar.
- Culminación y cumplimiento de los requisitos.
- Documentación final.

Después de definir el ciclo de vida, los roles con sus actividades dentro de cada una de sus fases, se muestra en la tabla 4, cuales son los artefactos que se deben desarrollar por fase. En la parte de la izquierda contiene cada fase del ciclo de vida y a la derecha define que artefacto será utilizado en cada una de ellas.

Tabla 4. Mapeo entre ciclo de vida y artefactos.

FASE DEL CICLO DE VIDA	ARTEFACTO
Planificación	Product backlog
	Historias de usuario
Desarrollo	Sprint backlog
	Libro de arquitectura
	diseño
	Prueba de desarrollo
Culminación	Documentación del producto
	Documentación del usuario

Fuente. Autor del proyecto

Los dos primeros artefactos que se muestran, hacen parte de la fase de planificación, dado que para esta parte del proyecto se usaron artefactos de XP y Scrum, con el fin de lograr que MAF tenga un mejor orden y además de esto la planificación y desarrollo de los requerimientos sea más precisa (Khelladi, Bendraou, Baarir, Laurent, & Gervais, 2015). Los requerimientos del sistema son listados en el product backlog, los proyectos se realizan en una serie de iteraciones o sprints, al product backlog se le asigna una prioridad y se seleccionan las tareas que se desarrollaran en cada sprint, luego de crear el product backlog se describirá cada requerimiento tras realizar esto las tareas son removidas al sprint backlog.

En la fase de desarrollo, además de hacer el sprint backlog, se debe producir el libro de arquitectura y el diseño, con el fin de proporcionar una lista donde se observan los problemas que pueda presentar la arquitectura y las decisiones que se han tomado con respecto a diseños, modelos y códigos (Eclipse Foundation, 2012). El ultimo artefacto de la fase de desarrollo es la prueba de desarrollo, la cual será ejecutada en cada sprint o iteración y al final del proyecto con el fin de saber si los requisitos propuestos han sido cumplidos y de que no existe ningún error en ellos.

Luego se realiza la documentación del producto este artefacto proporciona documentación sobre el producto para el propietario. Se crea la documentación del producto para el beneficio de la división de marketing de una organización, el director del programa, y aquellas personas que deben evaluar el valor comercial de un sistema en particular.

Dentro de la fase de culminación se realiza los documentos que pueden ser utilizados por los usuarios finales de un sistema o

producto en particular. Este tipo de documentación normalmente está escrita de una manera que permite a los usuarios del sistema encontrar fácilmente la información que necesitan para utilizar el producto. Como lo son manuales de usuario, Tutoriales, Ayuda en línea e Instrucciones de instalación.

A continuación la tabla 5 especifica que rol desarrolla cada artefacto, con el fin de orientar las actividades que les corresponde desarrollar. A la izquierda se define el artefacto que se desarrollara y a la derecha el rol que será responsable de ejecutarlo.

Tabla 5. Mapeo entre artefactos y roles

ARTEFACTO	ROL
Product backlog	Cliente
Historia de usuario	Cliente
	Delegado
Sprint backlog	Cliente
	Delegado
	Desarrollador
Libro de arquitectura	Desarrollador
Diseño	Desarrollador
Prueba de desarrollo	Tester
Documentación del producto	Desarrollador
	Tester
	Facilitador
Documentación del usuario	Desarrollador
	Tester
	Facilitador

Fuente. Autor del proyecto

3.2 Evaluación del método ágil utilizando la herramienta de comparación y análisis (4-DAT)

Esta herramienta tiene cuatro dimensiones que proporcionan los criterios de evaluación para determinar el grado en que un método cumple con las características de los métodos ágiles de desarrollo de software desde diferentes perspectivas. Los elementos que se evalúan son: el alcance, la agilidad, los valores ágiles y el proceso de software (Qumer & Brian, 2006).

La primera dimensión hace referencia al alcance del método muestra que MAF, es iterativo e incremental, maneja grupos pequeños de desarrollo, sigue un estándar de rápida retroalimentación, características esenciales en un método ágil.

La segunda dimensión evalúa la agilidad y lo hace de manera cualitativa al observar la tabla 6, los resultados que arroja se concluye que MAF, cumple con un 80% de los requisitos establecidos.

El grado de agilidad (DA) depende de los términos flexibilidad (FY), velocidad (SD), eficiencia (LS), aprendizaje (LG) y adaptabilidad (RS). Para calcular la agilidad en esta dimensión Qumer y Henderson-Sellers proponen la siguiente fórmulaDA (Object) = (1/m)∑m DA(del inglés, Object, Phase or Practices) (Qumer & Henderson-Sellers, 2008).

MAF al ser un híbrido de los métodos ágiles XP, Srum y OpenUP posee prácticas que estas últimas también poseen, o al menos una de ellas. Las enumeradas con 1, 2, 3, 4, 6, 7 y 10 han sido tomadas de XP por lo que se han evaluado de igual manera que por Quemer y Henderson (2008).

De forma semejante la enumerada con 11 ha sido evaluada, con la diferencia de que pertenece a Scrum, mientras que la 14, 15 y 16 hacen parte de OpenUP.

Teniendo en cuenta que al realizar reuniones para controlar el Sprint terminado y planificar el siguiente se realiza un juego de planificación; se enumera con 1 si pertenece

tanto a XP como a Scrum. Por otra parte se consideran a las prácticas 6, 9 y 14 nativas de MAF.

Dando oportunidad de proponer funcionalidades nuevas al sistema o sencillamente cambios significativos que el delegado no ha sugerido se calificó con 5, los criterios de flexibilidad, velocidad, eficacia, el equipo aprende con el aumento de la experiencia y perfectamente posee adaptabilidad.

Al sacar un producto funcional aunque con un número importante de requisitos por cumplir se poseen criterios de usuarios finales de manera rápida; pudiendo el equipo retroalimentarse a una mayor velocidad.

Tabla 6. Evaluación y comparación con la segunda dimensión

		Características de la agilidad.					
MAF		FY	SD	LS	LG	RS	TOTAL
I Fases							
1	Planificación	1	1	1	0	1	4
2	Desarrollo	1	1	0	1	1	4
3	Culminación	1	0	1	1	1	4
Total		3	2	2	2	3	12
Grado de agilidad		3/3	2/3	2/3	2/3	3/3	12/(5*3)
II Practicas							
1	Desarrollo iterativo e incremental mediante iteraciones cortas	1	1	1	1	1	5
2	Programación en parejas	0	0	0	0	0	0
3	Pruebas	0	1	1	1	1	4
4	40 horas semanales	1	1	1	1	1	5
5	Rápida retroalimentación	1	1	1	1	1	5
6	Diseño simple	0	1	1	1	1	4
7	Refactorización	0	0	0	0	0	0
8	Participación activa de todos los miembros del producto	1	1	1	1	0	4
9	Reutilización constante	0	0	0	0	0	0
10	Estándar de codificación	0	0	0	0	0	0
11	Reunión para examinar la iteración terminada y planificar la siguiente	1	0	1	1	1	4
12	Reportando progreso	1	1	1	1	1	5
13	Cliente en dentro del equipo	1	0	0	1	0	2
14	Uso de artefactos exclusivos para la arquitectura	0	1	1	1	1	4
15	Uso de artefactos exclusivos para el diseño	1	1	1	1	1	5
16	Documentación especifica	1	1	1	1	1	5
Total		9	10	11	12	10	52
Grado de agilidad		9/16	10/16	11/16	12/16	10/16	52/80

Fuente. Autor del proyecto

La tercera dimensión califica los valores agiles MAF describe una tarea para cada ítem establecido por 4-DAT, principios tales como: Individuos e iteraciones por encima

de procesos y herramientas, Software activo sobre documentación excesiva (Bennett, McRobb, & Farmer, 2006), la colaboración con el cliente más que la negociación de un contrato, responder a los cambios más que seguir estrictamente un plan son soportados por MAF.

Por último se tiene la dimensión que analiza los procesos de software y al ser comparado con XP, Scrum y OpenUP se puede ver que MAF cumple y tiene mejores características que los métodos mencionados, características tales como Product backlog, Historias de usuario, Iteraciones rápidas, Cliente en el equipo de desarrollo, todo en conjunto para su proceso de desarrollo, además de contar con la documentación del producto para su proceso de configuración del software.

Al realizar la comparación de XP, Scrum y OpenUP con el método creado MAF y evaluarlo con la herramienta 4-DAT para medir el grado de agilidad de este último, se puede concluir que es un método ágil, que posee características que hacen que logre ser una herramienta ágil en el desarrollo de proyectos de software, posee tres fases y una lista de artefactos que hacen que el método sea flexible, veloz y adaptable para proyectos de gestión de software. Estos resultados coinciden con los obtenidos por Qumer & Henderson-Sellers (2008), en donde con la misma herramienta se evaluaron el grado de agilidad en seis métodos ágiles seleccionados y dos métodos tradicionales (cascada y espiral). Llegando a la conclusión que el grado de agilidad depende también de la forma o los elementos con se construya el nuevo método.

4. CONCLUSIONES

Los métodos agiles tienen características y estructura similar, todas ellas encaminadas a organizar el desarrollo de software de forma sencilla. Proponer un nuevo método ágil, debe cumplir con los postulados del manifiesto ágil y vincular elementos de diseño más estructurados que los utilizados en los métodos actuales; es así que MAF, posee fases, roles y artefactos de XP, Scrum y OpenUP, integradas en una nueva propuesta.

Crear un nuevo método ágil que tome los elementos potenciadores o distintivos de otros métodos, garantiza la mejor adaptación y completitud de las fases, roles y artefactos que actualmente se llevan. La poca documentación que ofrecen los métodos actuales, genera que las decisiones de diseño y la arquitectura en general no existan o sean superficiales. Se debe agregar elementos nuevos y complementarios a los métodos actuales que eviten los problemas que genera esta falencia.

MAF, se considera ágil a pesar de agregar nuevos elementos y actividades a las que ofrecen los tres métodos trabajados; gracias a que cumple con los elementos que valida la herramienta de análisis y comparación 4-DAT. .

5. BIBLIOGRAFÍA

A. Qumer, B. H.-S. (2007). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. ScienceDirect. 1-16.

Albaladejo, X. (2010). Proyectos agiles. Recuperado el 18 de Julio de 2015, de http://www.proyectosagiles.org/desarrollo-iterativo-incremental

Anaya, R. (2006). Una visión de la enseñanza de la Ingeniería de Software como apoyo al mejoramiento de las empresas de software. REVISTA Universidad EAFIT, 42(141), 60-76.

Artefactos. (s.f.). Obtenido de http://clases3gingsof.wikifoundry.com/page/Artefactos

Bennett, S., McRobb, S., & Farmer, R. (2006). Analisis y Diseno Orientado a Objetos de Sistemas. Madrid: McGraw-Hill.

Deemer, P., Benelfield, G., Larman, C., & Vodde, B. (2012). The Scrum Primer A Lightweight Guide to the Theory and Practice of Scrum Version 2.0. Obtenido de http://assets.scrumfoundation.com/downloads/1/scrumprimer20.pdf?1352449266

Eclipse Foundation. (1 de Junio de 2012). OpenUP. Recuperado el 18 de Julio de 2015, de http://epf.eclipse.org/wikis/openup/

Elvesæter, B., & Benguria, G. (2013). A comparison of the Essence 1.0 and SPEM 2.0 specifications for software engineering methods. PMDE '13 Proceedings of the Third Workshop on Process-Based Approaches for Model-Driven Engineering (págs. 1-10). Monpellier: ACM.

Extreme Programing. (s.f.). Obtenido de www.extremeprograming.org

Hadar, I., & Sherman, S. (2012). Agile vs. plan-driven perceptions of software architecture. CHASE 12 Proceedings of the 5th International Workshop on Co-operative and Human Aspects of Software Engineering (págs. 50-55). Zurich: IEEE Press.

Highsmith, J. (2002). Agile Software Development Ecosystems. Addison-Wesley.

Kendall, K. E., & Kendall, J. E. (2005). Análisis y diseño de sistemas. Sexta edición. México: Pearson Educación.

Kent, B. (1999). Extreme Programming Explained: Embrace Change. EEUU: Addison-Wesley.

Khelladi, D.-E., Bendraou, R., Baarir, S., Laurent, Y., & Gervais, M.-P. (2015). A framework to formally verify conformance of a software process to a software method. SAC '15 Proceedings of the 30th Annual ACM Symposium on Applied Computing (págs. 1518-1525). Salamanca: ACM.

Kroll, P., & MacIsaac, B. (2006). Agility and Discipline Made Easy: Practices from OpenUP and RUP. United States of America: Pearson Education.

Maurer, F., & Melnik, G. (2006). Agile methods: moving towards the mainstream of the software industry. ICSE 06 Proceedings of the 28th international conference on Software

engineering (págs. 1057-1058). Shanghai: ACM.

Mendes Calo, K., Estevez, E. C., & Fillottrani, P. R. (2009). Un framework para evaluación de metodologías ágiles. 1-10. San Salvador de Jujuy, Jujuy, Argentin.

Nazareno, R., Leone, H., & Gonnet, S. (2013). Trazabilidad de procesos ágiles: un modelo para la trazabilidad de procesos scrum. XVIII Congreso Argentino de Ciencias de la Computación (págs. 920-929). Bahía Blanca: Red de Universidades con Carreras en Informática (RedUNCI).

Pillat, R. M., Oliveira, T. C., & Fonseca, F. L. (2012). Introducing software process tailoring to BPMN: BPMNt. ICSSP '12 Proceedings of the International Conference on Software and System Process. Zurich.

Pressman, R. (2006). Ingenieria del software: Un enfoque práctico. . México: McGraw-Hill.

Qumer, A., & Brian, H.-S. (2006). Measuring agility and adptability of agile methods: A 4-Dimensional Analytical tool. IADIS International Conference Applied Computing, 503-507.

Qumer, A., & Henderson-Sellers, B. (Marzo de 2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. In: Information and Software Technology. Information and Software Technology, 50(4), 280-295.

Sharp, H., Biddle, R., Gray, P., Miller, L., & Patton, J. (2006). Agile development: opportunity or fad? CHI EA '06 CHI '06 Extended Abstracts on Human Factors in Computing Systems. Montréal.

Tinoco, O., Rosales Lopez, P. P., & Salas Bacalla, J. (2010). Criterio de seleccion de metodologias de desarrollo de software. Red de revistas cientificas de America Latina y el Caribe, España y Portugal , 70-74.

VersionOne. (2014). Version one. Recuperado el 18 de Julio de 2015, de http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf

VERSIONONE, Scrum y XP . (2010). Recuperado el 2014, de http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf