ESTUDIO DE METODOLOGIAS ÁGILES PARA PROYECTOS DE SOFTWARE EN CORTO TIEMPO

STUDY METHODOLOGY FOR AGILE SOFTWARE PROJECT IN SHORT TIME

Ana Celmira Gualteros Gualteros
anagualteros@gmail.com
Ingeniera en telemática. Universidad Distrital Francisco José de Caldas.
Bogotá (Colombia)

Diana Paola Orjuela Escobar
paolin92@hotmail.com
Ingeniera de Sistemas. Fundación Universitaria Los Libertadores.
Bogotá (Colombia)

Tipo de Articulo: Reflexión

Fecha de recepción
Septiembre 24 de 2013
Fecha de Aceptación
Octubre 10 de 2013

ABSTRACT

On software develop projects depending on his complexion or dimension, it´s necessary apply one or many types of methodology structure o agile, agree with his characteristics and project nature, the project manager or leader have to stay on the capability, his knowledge and experience to adapt the better methodology that shape and ensure quality and efficient process.

Structure methodology drive into a standard and rigorous documentation for a system and be very helpful because offer a develop macro for software developers with less experience, for that reason the methodology it´s longer and spend more time besides give hardiness and support to process. The agile methodology allow to focus on develop and verification of software instead design and documentation, these methodology support an incremental vision for the specification, develop and software delivery. It´s more appropriate when the requirement requires flexibility

permanent in develop change and has the intention of delivery quickly functionality and while each functionality delivery it´s refine, the better for the client is get results with effective functionality

Key Word

Efficiency, software develop, methodology, project.

RESUMEN

Dependiendo de su complejidad y dimensión, se hace aplicable una o varias de los tipos de metodología ya sea estructuradas o ágiles; de acuerdo a sus características y naturaleza del proyecto, el líder o gerente de proyecto debe estar en la capacidad según su conocimiento y experiencia, de adaptar la metodología que mejor se adapte y desarrolle este asegurando procesos con calidad y eficiencia.

Las metodologías estructuradas conducen a documentación estandarizada y rigurosa para un sistema y son muy útiles al ofrecer un marco de desarrollo para los creadores de software con menor experiencia, por eso estas metodologías se hacen más largas en cuanto al consumo de tiempo, pero también es cierto que brindan robustez y soporte a los procesos. Por otro lado las metodologías ágiles permiten enfocarse más en el desarrollo y verificación del software en lugar del diseño y la documentación, estas metodologías se apoyan en el enfoque incremental para la especificación, el desarrollo y la entrega de software. Son más apropiados cuando los requerimientos requieren de flexibilidad y cambios permanentes durante el desarrollo y tienen la intención de entregar con prontitud las funcionalidades y a medida de cada entrega se va refinando e incrementando esas funcionalidades, pero lo mejor para el cliente es que va adquiriendo resultados con funcionalidades eficaces.

Palabras clave:

Eficiencia, desarrollo de software, metodología, proyecto.

I. INTRODUCCIÓN

Durante las últimas décadas, las notaciones de modelado y posteriormente las herramientas se concibieron determinantes para el éxito en el desarrollo de software, sin embargo, las expectativas no se cumplieron en su totalidad. Esto se debe en gran parte a que otro importante

elemento, la metodología de desarrollo, no había sido tenido en cuenta. De nada sirven buenas notaciones y herramientas si no se proveen directivas para su aplicación.

Así, esta década ha comenzado con un creciente interés en metodologías de desarrollo. Hasta hace poco, en la ingeniería de software las metodologías de desarrollo se planteaban de una forma tradicional, en la que se requiere de procesos rigurosos de planeación del proyecto, aseguramiento de calidad formalizada y uso de métodos y análisis de diseño apoyado por herramientas rigurosas y controladas con un modelo secuencial para realizar las actividades, por ejemplo que no se puede iniciar la construcción del software hasta tanto no se tenga la especificación de todos los requerimientos. Este esquema "tradicional" para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de ceremonia en el proceso. Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo unos niveles óptimos de calidad.

En este escenario, las metodologías ágiles emergen como una posible respuesta para llenar ese vacío metodológico. Por estar especialmente orientadas para proyectos de corto plazo, las metodologías ágiles constituyen una solución a la medida para ese entorno, aportando una elevada simplificación en los procesos y documentación, que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.

Al hablar de metodologías ágiles para el desarrollo de software, se hace referencia a un conjunto de métodos que enfatizan el enfoque iterativo e incremental, la adaptabilidad del proceso y la colaboración. Los métodos ágiles se caracterizan además por el hecho de que reducen la documentación y los procedimientos al mínimo.

Las metodologías ágiles son sin duda uno de los temas recientes en ingeniería de software con sus inicios en la década de 1990, donde varios desarrolladores propusieron nuevos ¨métodos agiles¨, que están acaparando gran interés. Es tal su impacto que actualmente ya es un área con cabida en prestigiosas revistas internacionales. En la comunidad de la ingeniería del software, se está viviendo

con intensidad un debate abierto entre los partidarios de las metodologías tradicionales (referidas peyorativamente como "metodologías pesadas") y aquellos que apoyan las ideas emanadas del "Manifiesto Ágil", un documento en el cual acordaron muchos de los desarrolladores líderes de estos métodos y el cual resume la filosofía "ágil".

El presente artículo presenta un estudio y análisis de las principales características de este tipo de metodologías y sus ventajas dentro del proceso de desarrollo de software.

1. MÉTODOS ÁGILES

En la década de 1980 predominaban como metodologías de desarrollo, las de tipo estructurado, las cuales eran difundidas por los desarrolladores como las más adecuadas para lograr un mejor software, mediante una cuidadosa planeación y estricto seguimiento de procesos controlados y documentados completamente. Esta percepción proviene de una tradicional comunidad de ingeniería de software, responsable del desarrollo de grandes sistemas de larga duración. En estas metodologías se enfoca más y se consume mayor atención y tiempo sobre los procesos de análisis y diseño más que en el desarrollo y pruebas del sistema.

Pero gracias al mundo cambiante en el cual se vive actualmente, el enfoque se fue inclinando hacía el desarrollo de sistemas pequeños y medianos que requieren de la simplificación de los procesos y optimizando los recursos y una de las características más destacadas es por realizar entregas incrementales. Por esto que en la década de 1990 un grupo de desarrolladores consientes de las limitantes en esta época y de la necesidad de abrir las posibilidades para desarrollar software con calidad y en periodos más cortos de tiempo, plasmaron en un documento llamado ¨Manifiesto Ágil¨ los doce principios, los cuales son:

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor. Este principio aporta y promueve que en los proyectos de desarrollo la retroalimentación sea continua con el cliente, así hay una mayor refinación del sistema y se va evidenciando el avance y la funcionalidad del sistema.

II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva. Los cambios en los requisitos deben verse

como algo positivo, ya que permiten aprender más, a la vez que logran una mayor satisfacción del cliente. Este principio también implica que el sistema debe caracterizarse por ser flexible para poder adecuarse a los cambios.

III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas. Las entregas al cliente de un producto es a la que realmente el cliente le interesa, para este no es de su prioridad los documentos de planificación y diseño, él lo que requiere es la funcionalidad que mejore sus procesos.

IV. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto. El proceso de comunicación y retroalimentación hace más efectivo el entendimiento de la necesidad.

V. Construir el proyecto en torno a individuos motivados. Las personas son el activo más importante dentro del proyecto de desarrollo, si el personal se desmotivado no se logrará con el desarrollo del ingenio y consecución de los objetivos.

VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo. Es la principal herramienta de comunicación asertiva y efectiva.

VII. El software que funciona es la medida principal de progreso. El estado de un proyecto no viene dado por la documentación generada o la fase en la que se encuentre, sino por el funcionamiento acertado.

VIII. Los procesos ágiles promueven un desarrollo sostenible. No se trata de desarrollar lo más rápido posible, sino de mantener el ritmo de desarrollo durante toda la duración del proyecto, asegurando en todo momento la calidad.

IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad. Producir código claro, complejo y estandarizado es un factor que apoya para que el desarrollo sea más rápido en el proyecto.

X. La simplicidad es esencial. Utilizar caminos más simples que sean consistentes con los objetivos

perseguidos. Entre más sencillo pero con calidad sea el código es más fácil adaptarlo a los cambios.

XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos. El equipo es informado de las responsabilidades y éstas recaen sobre todos sus miembros.

XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento. Gracias a que el entorno está cambiando continuamente, el equipo también debe ajustarse los nuevos escenarios de forma permanente.

Los firmantes de los valores y principios de este Manifesto son: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland y Dave Thomas. [1]

En este manifiesto plasmado por los autores descritos anteriormente, también describen los valores principales de esta metodología:

- A los individuos y las iteraciones sobre los procesos y las herramientas: Si el equipo funciona es más fácil conseguir el objetivo final. El personal es el principal factor de éxito de un proyecto de software.
- Al software operativo sobre la documentación exhaustiva: Al cliente le interesa y beneficia un sistema que funcione y no uno que falle y esté completamente documentado. Esta documentación debe ser corta y centrarse en lo fundamental, con esto se invierte tiempo en actividades más criticas
- La colaboración con el cliente sobre la negociación del contrato: se propone que exista una interacción constante entre el cliente y el equipo de desarrollo, así todo el equipo estará encaminado al mismo objetivo y enterado de cualquier cambio
- La respuesta al cambio sobre el seguimiento de un plan: En los proyectos se debe tener una habilidad de responder a los cambios ya sea por cualquier factor económico, tecnológico, etc y más aun con el mundo cambiante al que se enfrenta hoy en día la sociedad. Dos autores de la Universidad Politécnica de

Valencia (Patricio Letelier y Ma. Carmen Penadés) proponen que: ¨Una buena estrategia es hacer planificaciones detalladas para unas pocas semanas y planificaciones mucho más abiertas para unos pocos meses¨ [2].

En las metodologías agiles en el desarrollo de software se consideran el diseño y la implementación como las actividades centrales en el proceso de software y en metodologías tradicionales estructuras basadas en un plan, la iteración ocurre dentro de las actividades con documentos formales usados para comunicarse entre etapas del proceso, por ejemplo dentro de estas, los requerimientos evolucionarán y al final se producirá una especificación detallada de ellos y estos a este entonces serán una entrada al proceso de diseño en cambio en las ágiles la iteración ocurre a través de las actividades, por tanto los requerimientos y el diseño se desarrollan en conjunto.

A continuación en la figura No. 1 se describe la diferencia entre una especificación ágil y la dirigida por un plan.

Desarrollo basado en un plan

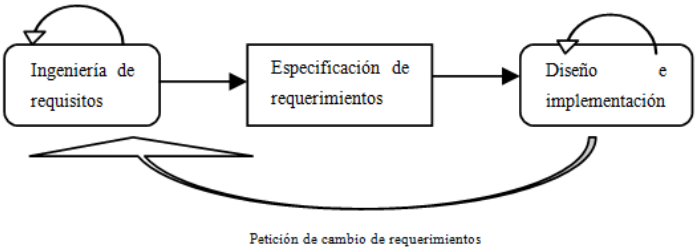


Figura 1. Especificación ágil y dirigida por un plan

2. METODOLOGÍAS ÁGILES PARA EL DESARROLLO DE SOFTWARE

2.1 Programación extrema (Extreme Programming XP)

Propuesta por Kent Beck (2000), esta se basa en un conjunto reducido de prácticas que reflejan una serie de valores como: comunicación donde se deben adoptar técnicas que la fomenten, simplicidad, que las soluciones sean caracterizadas por ser simples y efectivas, retroalimentación esta se aplica a través de las pruebas, coraje que implica no dudar y si se llega a requerir a cambiar el diseño y calidad.

La programación extrema se caracteriza por aplicar unas prácticas puntuales: a) los requerimientos se expresan como escenarios llamados historias de

usuario, que se implementan directamente como una serie de tarea, b) liberaciones pequeñas del sistema, en periodos más cortos de tiempo y se agregan incrementalmente funcionalidades, c) los desarrolladores trabajan en pares y cada uno comprueba el trabajo del otro, lo cual permite un mejor entendimiento y solución más completa, d) antes de escribir el código desarrollan pruebas para cada tarea, por lo cual en las pruebas integradas se ejecutan con un mayor éxito, e) propiedad colectiva, los desarrolladores en pares trabajan en todas las áreas, de manera que no se desarrollan islas ya que todos se responsabilizan del trabajo de todos y así se controla el riesgo en cuanto a cambios de personal, f) integración continua, a medida que se termina una tarea se va integrando al sistema al cual se le aplica las pruebas de unidad garantizando su funcionalidad e integración, g) ritmo sustentable, no se consideran aceptables grandes cantidades extras de tiempo ya que esto puede llegar a incurrir en baja calidad del código por el desgaste del equipo al que se lleva, y por último h) cliente en sitio, un representante del cliente (usuario final) tiene que disponer de tiempo completo dentro del equipo de desarrollo.

2.2 SCRUM

Es un marco de trabajo para la gestión y desarrollo de software, el cual se caracteriza por ser un proceso iterativo e incremental por lo cual corresponde a un método ágil (ver figura 2). Este método denomina 'sprints'[1] a las iteraciones. Estas iteraciones tienen la característica que son muy cortas, típicamente de dos a cuatro semanas. Cada una de estas se produce en una versión funcional y entregable del producto. El método se articula alrededor por tres roles, tres ceremonias y tres artefactos:

- ✓ Roles: propietario del producto, el gestor SCRUM que es el responsable de que el equipo sea funcional y productivo y los integrantes del equipo de desarrollo que se caracterizan por son capaces de auto-organizarse.

- ✓ Ceremonias: es la reunión de planificación del siguiente 'sprint' (entrega), las reuniones SCRUM diarias de seguimiento y control y las reuniones de revisión.

- ✓ Artefactos: es la bitácora del producto que es la lista priorizada de requisitos, la bitácora del 'sprint' que contiene la planificación de las tareas de la

[1] Sprint es el período en el cual se lleva a cabo el trabajo en sí

siguiente iteración y el diagrama de progreso que se utiliza para el seguimiento de los 'sprints'.

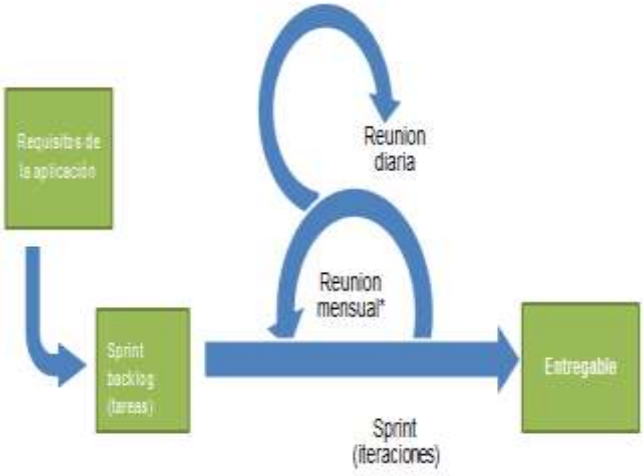
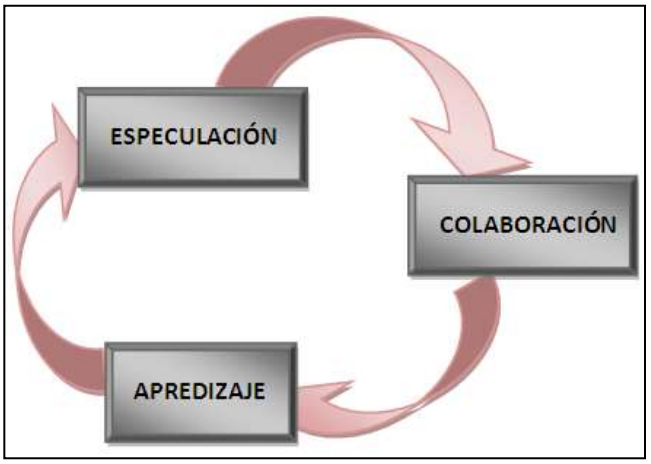


Figura 2. Procesos del método SCRUM

2.3 Desarrollo Adaptativo de Software (DAS)

Este método fue propuesto por Jim Highsmith como una técnica para el desarrollo de software y sistemas compuestos. Los principios filosóficos del DAS se centran en el trabajo en equipo y su organización propia.

Highsimth plantea una especie de ciclo de vida del DAS (Desarrollo Adaptativo de Software) que integra tres fases: especulación, colaboración y aprendizaje. Véase la figura 3.



Especulación:

- ✓ Planeación adaptativa del ciclo
- ✓ Enunciado de la misión de los clientes
- ✓ Restricciones del proyecto
- ✓ Requerimientos básicos
- ✓ Plan de entrega en el tiempo

Colaboración:

- ✓ Recabar requerimientos
- ✓ Miniespecificaciones

Aprendizaje:

- ✓ Componentes implementados o probados
- ✓ Grupos de enfoque para recibir retroalimentación
- ✓ Revisiones técnicas formales

La filosofía DAS (Desarrollo Adaptativo de Software) hace especial énfasis en el desarrollo adaptativo enfocado en la dinámica de los equipos con organización propia, la colaboración interpersonal y el aprendizaje individual y del equipo que generan equipos para proyectos de software que tienen una probabilidad de éxito mucho mayor.

2.4 Método de Desarrollo de Sistemas Dinámicos (MDSD)

Este método es un enfoque de desarrollo ágil de software que brinda una estructura para desarrollar y dar mantenimiento a sistemas que cumplan restricciones apretadas de tiempo a través de la realización de prototipos incrementales en un ambiente controlado de proyectos.

El MDSD (Método de Desarrollo de Sistemas Dinámicos) es un proceso iterativo de software en el que cada iteración sigue el principio de la regla de Pareto: El 80% de una aplicación puede entregarse en el 20% del tiempo que tomaría entregarla completa (100%). Par este método de ha definido un modelo de proceso ágil, llamado ciclo de vida MDSD (Método de Desarrollo de Sistemas Dinámicos), que define tres ciclos iterativos distintos, precedidos de dos actividades adicionales al ciclo de vida:

- ✓ Estudio de factibilidad: Establece los requerimientos y restricciones básicas del negocio.
- ✓ Estudio del negocio: Establece los requerimientos e información funcionales
- ✓ Iteración del modelo funcional: Produce un conjunto de prototipos incrementales que demuestran al cliente la funcionalidad.
- ✓ Diseño e iteración de la construcción: Revisa los prototipos construidos durante la interacción
- ✓ Implementación: Coloca el incremento más reciente del software (un prototipo “operacional”) en el ambiente de operación.

El MDSD (Método de Desarrollo de Sistemas Dinámicos), se combina con XP para dar un enfoque de combinación que define un modelo sólido del proceso

(ciclo de vida MDSD) con las prácticas detalladas (XP) que se requieren para elaborar incrementos de software.

2.5 Cristal

Es una familia de modelos de proceso ágiles creada por Alistar Cockburn, a fin de obtener un enfoque de desarrollo de software que premia la "maniobrabilidad" durante un "juego" cooperativo con recursos limitados, de invención y comunicación, a fin de entregar software útil que funcione y con el objetivo secundario de plantear el siguiente "juego".

Para lograr la maniobrabilidad, Cockburn y Highsmith definieron un conjunto de metodologías, cada una de ellas con elementos fundamentales comunes a todos, y roles, patrones de proceso, producto del trabajo y prácticas que son únicas para cada uno. La familia Cristal en realidad es un conjunto de ejemplos de procesos ágiles que han demostrado ser efectivos para diferentes tipos de proyectos.

2.6 Desarrollo Impulsado por las Características (DIC)

Fue concebido originalmente por Peter Coad como un modelo práctico de proceso para la ingeniería de software orientada a objetos.

Este modelo adopta los siguientes principios:

- ✓ Hace énfasis en el trabajo en equipo
- ✓ Administra la complejidad de los problemas y del proyecto con el uso de la descomposición basada en las características.
- ✓ Comunica los detalles técnicos en forma verbal, gráfica y con medios basados en texto.
- ✓ Se da prioridad en las actividades de aseguramiento de la calidad del software mediante la estrategia de desarrollo incremental, el uso de inspecciones del diseño y del código, la aplicación de auditorías de aseguramiento de la calidad del software, el conjunto de mediciones y el uso de patrones.

El enfoque DIC (Desarrollo Impulsado por las Características) define cinco actividades estructurales o procesos, como se muestra en la figura 4.

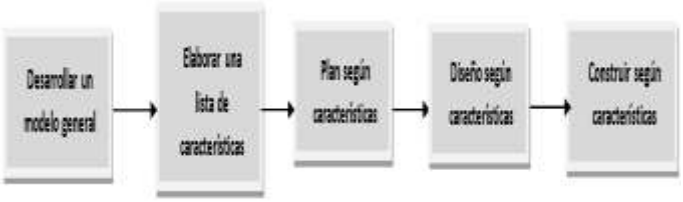


Figura 4. Desarrollo Impulsado por las Características

2.7 Modelado Ágil (MA)

El modelado ágil (MA) es una metodología basada en la práctica para modelar y documentar con eficacia los sistemas basados en software. Dicho de otro modo, es un conjunto de valores, principios y prácticas para hacer modelos de software aplicables de manera eficaz y ligera a un proyecto de desarrollo de software. Los modelos ágiles son más eficaces que los tradicionales porque son sólo buenos, sin pretender ser perfectos.

El modelado ágil adopta todos los valores del manifiesto ágil, y sugiere una amplia variedad de principios de modelado ”fundamentales” y “suplementarios”, pero algunos con exclusivos de este modelado, los cuales se mencionan a continuación:

- ✓ Modelo con un propósito: Un desarrollador que use modelado ágil debe tener en mente una meta específica antes de crear el modelo.
- ✓ Uso de Modelos Múltiples: El Modelado Ágil sugiere que para dar la perspectiva necesaria, cada modelo debe presentar un diferente aspecto del sistema y que sólo deben utilizarse aquellos modelos que den valor al público al que se dirigen.
- ✓ Viajar ligero: Conforme avanza el trabajo de ingeniería de software, conserve sólo aquellos modelos que agreguen valor a largo plazo y elimine los demás.
- ✓ El contenido es más importante que la representación: El modelado debe transmitir información al público al que se dirige.
- ✓ Conocer los modelos y herramientas que se utilizan en su creación: Entender las fortalezas y debilidades de cada modelo y las herramientas que se emplean para crearlos.
- ✓ Adaptación Local: El enfoque de modelado debe adaptarse a las necesidades del equipo ágil.

2.8 El Proceso Unificado Ágil (PUA)

El proceso unificado ágil (PUA) adopta una filosofía "en serie para lo grande" e "iterativa para lo pequeño" a fin de construir sistemas basados en computadora. Cada iteración del proceso unificado ágil aborda las actividades siguientes:

- ✓ Modelado: Se crean representaciones de UML[2] de los dominios del negocio y el problema.
- ✓ Implementación: Los modelos se traducen a código fuente.
- ✓ Pruebas: Igual que como sucede con la metodología XP (Extreme Programming), el equipo de trabajo diseña y ejecuta una serie de pruebas para detectar errores y garantizar que el código fuente cumple sus requerimientos.
- ✓ Despliegue: Se centra en la entrega de un incremento de software y en la obtención de retroalimentación de los usuarios finales.
- ✓ Configuración y administración del proyecto: En este contexto se incluye la administración del cambio y el riesgo, y el control de cualquier producto del trabajo que produzca el equipo.

3. DIFERENCIAS ENTRE METODOLOGÍAS ÁGILES Y NO ÁGILES

De los diversos autores que se han estudiado sobre estas metodologías, se ha resumido en la tabla 1 las diferencias que se pueden identificar las metodologías ágiles de las tradicionales:

Metodologías Ágiles Metodologías Tradicionales	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Tabla 1. Diferencias entre metodologías ágiles y no ágiles

[2] UML (Unified Modeling Languaje): Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software

Estas marcadas diferencias han hecho que las metodologías ágiles hoy en día sean tan estudiadas y utilizadas en la ingeniería de software, ya que están supliendo vacíos en los proyectos emergentes que nacen día a día en las organizaciones, ya que los negocios y procesos requieren de tiempos record.

II. CONCLUSIONES

En las metodologías estudiadas en el presente artículo, se evidencian cuatro características comunes a todas ellas: la simplicidad, la agilidad, el trabajo en equipo y la comunicación que deben ser predominantes a la hora de desarrollar software.

El enfoque iterativo que proporciona un proceso ágil, que le permite a los clientes evaluar el incremento del software de manera continua, brindado la retroalimentación necesaria al equipo de trabajo. Se puede evidenciar su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Las metodologías ágiles permiten una mayor flexibilidad que las metodologías tradicionales de desarrollo, y son capaces de ajustarse a las cambiantes necesidades de los clientes, del mercado y de los desafíos imprevistos que plantea la tecnología.

Los objetivos que deben primar en cualquier desarrollo de software son: buscar la satisfacción del cliente, la entrega temprana de software con calidad y una simplicidad general del desarrollo.

Una metodología ágil prioriza los aspectos de desarrollo de alto riesgo permitiendo así una reducción de los riesgos iniciales.

III. REFERENCIAS

[1] J. José H. Cános, Patricio Letelier, Ma. Carmen Penadés, "Métodologias Ágiles en el Desarrollo de Software," DSIC Universidad Politécnica de Valencia

[2] Ian Sommerville, Ingeniería de Software, Pearson Educación, México, 2011

[3] Salvador Sánchez, Miguel Ángel Sicilia, Daniel Rodríguez, Ingeniería del Software 'Un enfoque desde la guía SWEBOK', ALFAOMEGA, México

[4] Roger S Pressman, Ingeniería del software 'Un enfoque práctico', Séptima Edición, Mac Graw Hill, 2010.