

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/334000094>

ANÁLISIS COMPARATIVO DE METODOLOGÍAS ÁGILES DE DESARROLLO DE SOFTWARE: UNA REVISIÓN BIBLIOGRÁFICA

Chapter · January 2019

CITATION

1

READS

10,838

4 authors, including:



Juan Casierra Cavada

PUCESE

6 PUBLICATIONS 9 CITATIONS

[SEE PROFILE](#)



Xavier Quiñonez-Ku

Pontificia Universidad Católica del Ecuador Sede Esmeraldas

12 PUBLICATIONS 35 CITATIONS

[SEE PROFILE](#)



Luis Herrera-Izquierdo

5 PUBLICATIONS 9 CITATIONS

[SEE PROFILE](#)

Análisis comparativo de metodologías ágiles de desarrollo de software: una revisión bibliográfica

Comparative analysis of agile software development methodologies: a bibliographic review

Xavier Quiñónez-Ku, Juan Casierra Cavada, Luis Herrera-Izquierdo, Junior Mera Quiroz

Carrera de Sistemas y Computación, Pontificia Universidad Católica del Ecuador Sede Esmeraldas,
xavier.quinonez@pucese.edu.ec,

Carrera de Sistemas y Computación, Pontificia Universidad Católica del Ecuador Sede Esmeraldas,
juan.casierrac@pucese.edu.ec,

Professional Services, ThoughtWorks EC, lherrera@thoughtworks.com,
LatAm Autos, jmera@latamautos.com.

Resumen— El concepto de metodologías ágiles ha generado un gran interés a las organizaciones para lograr objetivos de desarrollo de software rápidos y funcionales, gracias a sus características de concentrarse más en la entrega de un software funcional en lugar de presentar una carga masiva de documentación; respuesta rápida a los cambios en los requisitos en lugar de seguir un plan prescrito; colaboración con clientes en lugar de negociación de contratos y dando más preferencia a individuos e interacciones sobre procesos y herramientas. En esta investigación se realizó una comparativa entre las metodologías ágiles a través de una revisión bibliográfica, para ayudar en la selección de la metodología de desarrollo de software apropiada en un escenario particular.

Abstract-- The concept of agile methodologies has generated great interest for organizations to achieve fast and functional software development objectives, thanks to their characteristics of concentrating more on the delivery of a functional software instead of presenting a massive documentation load; quick response to changes in requirements instead of following a prescribed plan; collaboration with clients instead of negotiating contracts and giving more preference to individuals and interactions about processes and tools. In this research, a comparison was made between the agile methodologies through a bibliographic review, to assist in the selection of the appropriate software development methodology in a particular scenario.

I. INTRODUCCIÓN

Es común encontrar personas cuyo criterio acerca del desarrollo de software está ligado únicamente a los programas de computadora, lo cual es bastante erróneo, ya que el desarrollo de

software también está relacionado con todos aquellos documentos que recaban información acerca de los requerimientos, el diseño y demás procesos vinculados [1].

El desarrollo de software no es una tarea sencilla, por mucho tiempo esta labor se ha llevado adelante sin una metodología definida. La disciplina que se encarga de determinar, analizar, describir y establecer las diferentes metodologías y lineamientos para mejorar u optimizar el desarrollo de sistemas informáticos se la denomina Ingeniería de Software [2].

Algunos autores definen una metodología como una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo. Sin embargo, una metodología es algo más que una colección, puesto que se basa en una filosofía, distinguiéndose de los métodos o de las simples recetas, que marcan simplemente unos pasos a seguir. Así, las metodologías difieren ya sea por la cantidad de fases, las técnicas de cada fase, el contenido de la fase o en su base filosófica, todo esto se aplica, dependiendo del contexto de desarrollo, tamaño del proyecto o del equipo de trabajo, cultura organizacional, entre otros aspectos [3].

Según [4], la complejidad de los sistemas actuales, los cambios repentinos en el contexto de estos sistemas y las modificaciones en los requerimientos del cliente una vez que se ha comenzado el desarrollo de un proyecto de software, generan un ambiente donde la planificación, el desarrollo, la administración y el control del mismo resultan difíciles de estimar o evaluar. Este tipo de escenarios requiere de metodologías de desarrollo de software que permitan generar resultados rápidamente.

II. HISTORIA DEL ARTE

Metodologías de desarrollo

Según [5], en la década de los noventa surgieron metodologías de desarrollo de software ligeras, más conocidas como metodologías ágiles, que buscaban reducir la probabilidad de fracaso por subestimación de costos, tiempos y funcionalidades en los proyectos de desarrollo de software. Estas metodologías nacieron como reacción a las metodologías existentes con el propósito de disminuir la burocracia que implica la aplicación de las metodologías tradicionales en los proyectos de pequeña y mediana escala. Las metodologías tradicionales buscan imponer disciplina al proceso de desarrollo de software y de esa forma volverlo predecible y eficiente. Para conseguirlo se soportan en un proceso detallado con énfasis en planeación propia de otras ingenierías. El principal problema de este enfoque es que hay muchas actividades que hacer para seguir la metodología y esto retrasa la etapa de desarrollo.

Diferentes investigadores comparan enfoques tradicionales y ágiles en sus diferentes perspectivas, en la Tabla I se describen las diferencias entre las metodologías ágiles de desarrollo y las metodologías tradicionales según [6]–[10].

TABLA I
DIFERENCIAS ENTRE METODOLOGÍAS TRADICIONALES Y ÁGILES

	Metodologías tradicionales	Metodologías ágiles
Hipótesis fundamental	Los sistemas son totalmente especificables, predecibles y se desarrollan a través de una planificación detallada y extendida.	El software adaptativo de alta calidad es desarrollado por pequeños equipos que utilizan el principio de mejora continua del diseño y las pruebas basadas en una rápida respuesta y cambio.
Estilo de gestión	Comando y control	Liderazgo y colaboración
Conocimiento administrativo	Explícito	Táctico
Comunicación	Formal	Informal
Modelo de desarrollo	Modelo de ciclo de vida (cascada, espiral o modelos modificados)	Modelo evolutivo de entrega
Estructura organizacional	Mecánico (burocrático, alta)	Orgánico (flexible y participativo,

	formalización), dirigido a grandes organizaciones.	fomenta la cooperación social), dirigido a pequeñas y medianas organizaciones.
Control de calidad	Planificación difícil y control estricto. Pruebas difíciles y tardías	Control permanente de requisitos, diseño y soluciones. Pruebas permanentes.
Requisitos de usuario	Detallado y definido antes de la codificación / implementación.	Entrada interactiva
Costo de reinicio	Alto	Bajo
Dirección de desarrollo	Fijo	Fácilmente cambiante
Pruebas	Después de completar la codificación	Cada iteración
Participación del cliente	Bajo	Alto
Habilidades adicionales requeridas de los desarrolladores	Nada en particular	Habilidades interpersonales y conocimientos básicos del negocio.
Escala apropiada del proyecto	Gran escala	Baja y mediana escala
Desarrolladores	Planificado, con capacidades adecuadas, acceso a conocimientos externos	Ágil, con conocimientos avanzados y cooperativos.
Clientes	Con acceso al conocimiento, cooperativo, representativo y empoderado	Dedicado, informado, cooperativo, representativo y capacitado.
Requerimientos	Muy estable, conocido de antemano	Emergente, con cambios rápidos.
Arquitectura	Diseño para requerimientos actuales y previsibles.	Diseño para requerimientos actuales.
Remodelación	Costoso	No es caro
Tamaño	Grandes equipos y proyectos	Pequeños equipos y proyectos
Objetivos principales	Alta seguridad	Valor rápido

Nota: La tabla muestra las principales diferencias entre las metodologías ágiles y las metodologías tradicionales

Metodologías ágiles de desarrollo

El enfoque de las metodologías ágiles está teniendo una amplia efectividad en proyectos donde los requisitos son muy cambiantes,

ya que en este tipo de proyectos la comunicación con el cliente debe ser fundamental y precisamente ese es uno de los principios básicos de las metodologías ágiles.

En marzo de 2001, 17 críticos de los modelos de producción basados en procesos resumieron en cuatro postulados lo que hoy en día se conoce como el Manifiesto Ágil [11]. Los cuatro postulados mencionados en aquella reunión mencionaban los siguientes criterios: Individuos e interacciones sobre procesos y herramientas, Software que funciona sobre documentación exhaustiva, Colaboración de clientes sobre la negociación del contrato y Respuestas a cambios sobre seguir un plan. Estos cuatro postulados son considerados como la definición canónica del desarrollo ágil.

A continuación, en la Figura 1 se puede evidenciar las principales razones por las que las organizaciones a nivel mundial están adoptando las metodologías ágiles de desarrollo de software:



Fig. 1. Razones para adoptar metodologías ágiles [12].

En la actualidad existen un sin número de metodologías ágiles, unas más populares que otras, cada una aportando al desarrollo ágil distintos métodos que ayudan a mejorar de una manera eficaz la calidad del software. Entre las metodologías ágiles más populares se encuentran las siguientes:

A. Programación Extrema (XP):

La Programación Extrema o XP (Extreme Programming) es un enfoque de desarrollo de sistemas que acepta lo que se conoce como buenas prácticas en esta área y las lleva al extremo [13], [14]. En esta metodología cabe resaltar la importancia del cliente, las pruebas, la

refactorización, la simplicidad, la propiedad colectiva del código que se ven reflejadas en las cuatro prácticas esenciales de XP [15], [16]:

- *Entregas limitadas o pequeñas:* Consiste en realizar entregas parciales de módulos del sistema.
- *Semana de trabajo de 40 horas:* Los equipos de desarrollo de XP trabajan de manera intensa durante una semana típica de 40 horas.
- *Cliente en el sitio:* Esta práctica insiste en que el cliente debe hacer parte fundamental y activa del grupo de trabajo y debe estar presente durante todo el proceso de desarrollo.
- *Programación en Pareja:* Con esto se busca aumentar la calidad del código, ahorrar tiempo, estimula la creatividad y la reducción de código fuente.

La programación extrema engloba un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades estructurales: planeación, diseño, codificación y pruebas. La Figura 2 muestra cada una de las actividades de XP, y resalta las tareas clave de cada una [13], [15], [17], [18]:

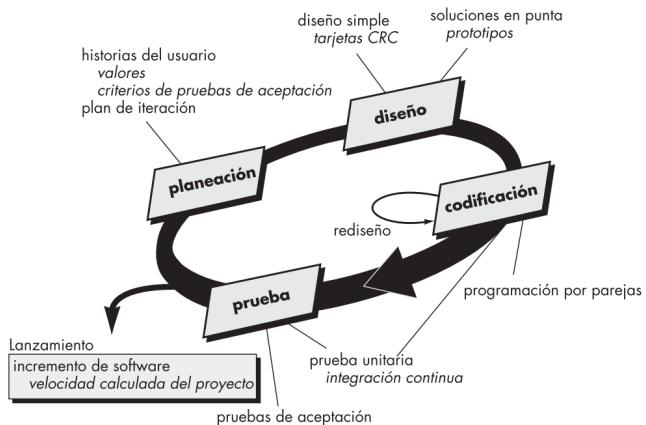


Fig. 2. El proceso de la programación extrema [15].

Planeación: Esta actividad comienza escuchando a los clientes, para entender el contexto del negocio y definir las características principales y funcionalidad que se requiere, estas características se transforman en requerimientos del negocio que se especifican mediante Historias de Usuario; las cuales recogen la interacción hablada entre desarrolladores y usuarios. Una vez hechas las Historias de Usuario, el equipo de desarrollo las divide en tareas, estima el esfuerzo, recursos requeridos para su implementación, se genera el plan de entregas, las iteraciones, la rotación de parejas y las reuniones diarias.

Diseño: Es la etapa en donde son evaluadas las historias de usuario por el equipo del proyecto para dividirlas en tareas, cada tarea representa una característica distinta del sistema y se puede

diseñar una prueba de unidad que verifique cada tarea, estas tareas se representan por medio de las tarjetas CRC (Clase-Responsabilidad-Colaborador). Las tarjetas CRC identifican y organizan las clases bajo el paradigma orientado a objetos (lo que incluye asignación de responsabilidades), cada tarjeta contiene el nombre de la clase (que representa una o más historias de usuario), una descripción de las responsabilidades o métodos asociados con la clase, así como la lista de las clases con que se relaciona o que colaboran con ella. Las tarjetas CRC son el único trabajo de diseño que se genera como parte del proceso de XP.

Codificación: Se lleva a cabo la programación en pareja, la unidad de pruebas y la integración del código. Durante esta etapa se espera la disponibilidad del cliente para que éste pueda resolver cualquier duda que se presente durante una jornada de trabajo.

Prueba: Cada tarea que se identificó con las historias de usuario, representa una característica distinta del sistema y se realiza una prueba de unidad por cada una de ellas, existen pruebas unitarias las cuales son diseñadas para probar cada uno de los métodos y clases, dichas pruebas son realizadas por los programadores.

B. Scrum:

Su nombre no corresponde a una sigla, sino a un concepto deportivo, propio del rugby, relacionado con la formación requerida para la recuperación rápida del juego ante una infracción menor [5], [19]. Scrum es un modelo de desarrollo ágil caracterizado por [11], [20], [21]:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos autoorganizados, que en la calidad de los procesos empleados.
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizarlas una tras otra.

Según [15], [22], los principios Scrum son congruentes con el manifiesto ágil y se utilizan para guiar actividades de desarrollo dentro de un proceso de análisis que incorpora las siguientes actividades estructurales: requerimientos, análisis, diseño, evolución y entrega. El flujo general del proceso Scrum se ilustra en la Figura 3:

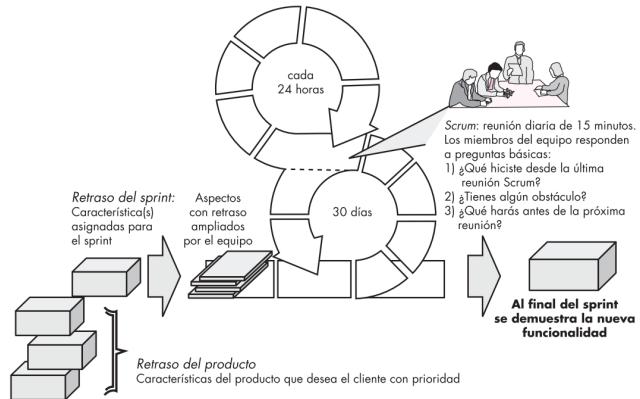


Fig. 3. Flujo del proceso Scrum [15].

Retraso: lista de prioridades de los requerimientos o características del proyecto que dan al cliente un valor del negocio.

Sprints: consiste en unidades de trabajo que se necesitan para alcanzar un requerimiento definido en el retraso que debe ajustarse en una caja de tiempo predefinida (lo común son 30 días).

Reuniones Scrum: son reuniones breves (de 15 minutos, por lo general) que el equipo Scrum efectúa a diario.

Demostraciones preliminares: entregar el incremento de software al cliente de modo que la funcionalidad que se haya implementado pueda demostrarse al cliente y éste pueda evaluarla.

Las características del desarrollo basado en Scrum son [10], [23]:

Colaboración: El desarrollo basado en Scrum promueve la colaboración ya que está impulsado por equipos multifuncionales donde cada persona con sus habilidades y experiencia contribuye a la mejor solución de diseño. Un equipo multifuncional incluye una combinación de programadores, arquitectos de software, analistas de software y expertos en control de calidad.

Reuniones diarias: La metodología Scrum está marcada por reuniones diarias de corta duración en las que el equipo de desarrollo de productos se comunica y evalúa el estado de progreso del desarrollo de software, lo que aumenta la productividad de los miembros del equipo.

Product Backlog: El product backlog captura los requisitos para que un producto de software se entregue con éxito. Mantiene una lista ordenada de características, correcciones de errores, requisitos no funcionales.

Sprint Backlog: El sprint backlog registra la lista de tareas que realizará el equipo de desarrollo durante el próximo sprint. Esta lista se elabora recogiendo las tareas desde la parte superior de la cartera de productos hasta que se realice el trabajo suficiente para el próximo sprint, teniendo en cuenta la capacidad de trabajo y los resultados pasados del equipo de desarrollo.

Roles: El desarrollo basado en Scrum se rige por 3 funciones principales [19], [24]:

- Product Owner: responsable de definir, priorizar y comunicar los requisitos del producto y guía el proceso de desarrollo del producto.

- Equipo de desarrollo: responsable de ejecutar las tareas asignadas por el propietario del producto dentro del plazo del sprint. Por lo general, un equipo multifuncional de 3 a 9 individuos implementa las tareas de desarrollo del producto previstas por el propietario del producto.
- Scrum Master: responsable de hacer cumplir las reglas y los principios del desarrollo basado en Scrum. El Scrum Master elimina los impedimentos para el desarrollo y ayuda a mejorar el proceso, el equipo de desarrollo y el producto de software que se está desarrollando.

C. Desarrollo adaptativo de software (ASD):

El desarrollo adaptativo de software (ASD) es una técnica para elaborar software y sistemas complejos. Los fundamentos filosóficos del ASD se centran en la colaboración humana y en la organización propia del equipo [15], [25], [26]. La Figura 4 ilustra el ciclo de vida del ASD:

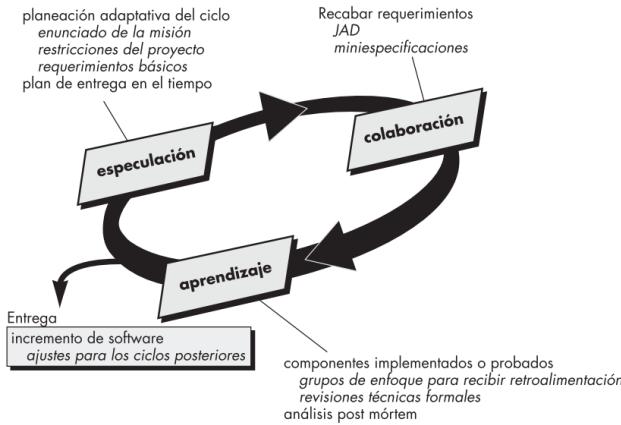


Fig. 4. Desarrollo adaptativo de software [15].

El “ciclo de vida” del ASD incorpora tres fases: especulación, colaboración y aprendizaje [15], [25], [27].

En la *especulación*, se inicia el proyecto y se lleva a cabo la planeación adaptativa del ciclo. La especulación emplea la información de inicio del proyecto para definir el conjunto de ciclos de entrega (incrementos de software) que se requerirán para el proyecto.

Las personas motivadas usan la *colaboración* de manera que multiplica su talento y producción creativa más allá de sus números absolutos. Este enfoque es un tema recurrente en todos los métodos ágiles. Sin embargo, la colaboración no es fácil. Incluye la comunicación y el trabajo en equipo, pero también resalta el individualismo porque la creatividad individual desempeña un papel

importante en el pensamiento colaborativo.

Conforme los miembros de un equipo ASD comienzan a desarrollar los componentes que forman parte de un ciclo adaptativo, el énfasis se traslada al *aprendizaje* de todo lo que hay en el avance hacia la terminación del ciclo.

D. Feature Drive Development (FDD):

El desarrollo impulsado por las características es una metodología ágil para el desarrollo de sistemas, basado en la calidad del software, que incluye un monitoreo constante del proyecto. Al igual que otros proyectos ágiles, adopta una filosofía que [15], [27], [28]:

- pone el énfasis en la colaboración entre los integrantes del equipo;
- administra la complejidad de los problemas y del proyecto con el uso de la descomposición basada en las características, seguida de la integración de incrementos de software, y
- comunica los detalles técnicos en forma verbal, gráfica y con medios basados en texto.

El desarrollo impulsado por las características se divide en 5 fases y es de tipo incremental, constando cada incremento (iteración) en 2 fases: diseño y construcción de una característica. En la Figura 5 se pueden distinguir estas fases:

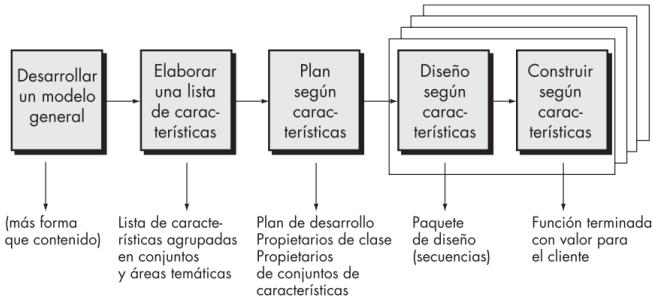


Fig. 5. Desarrollo impulsado por las características (Feature Drive Development (FDD)) [15], [25].

Se explica a continuación cada etapa [15], [25], [27]:

- *Desarrollar un modelo global:* Al inicio del desarrollo se construye un modelo teniendo en cuenta la visión, el contexto y los requisitos que debe tener el sistema a construir. Este modelo se divide en áreas que se analizan detalladamente. Se construye un diagrama de clases por cada área.
- *Construir lista de características:* Se elabora una lista que resume las funcionalidades que debe tener el sistema, cuya lista es evaluada por el cliente. Cada funcionalidad de la lista se divide en funcionalidades más pequeñas para un mejor entendimiento del sistema.
- *Planificar:* Se procede a ordenar los conjuntos de funcionalidades conforme a su prioridad y dependencia, y se asigna a los programadores jefes.
- *Diseñar:* Se selecciona un conjunto de funcionalidades de la lista. Se procede a diseñar y construir la funcionalidad

mediante un proceso iterativo, decidiendo que funcionalidad se van a realizar en cada iteración. Este proceso iterativo incluye inspección de diseño, codificación, pruebas unitarias, integración e inspección de código.

- *Construir:* Se procede a la construcción total del proyecto.

E. Dynamic Systems Development Method (DSDM):

El método de desarrollo de sistemas dinámicos (DSDM) es un enfoque de desarrollo ágil de software que “proporciona una estructura para construir y dar mantenimiento a sistemas que cumplan restricciones apretadas de tiempo mediante la realización de prototipos incrementales en un ambiente controlado de proyectos” [11], [15]. El DSDM es un proceso iterativo de software en el que cada iteración sigue la regla de 80 por ciento. Es decir, se requiere sólo suficiente trabajo para cada incremento con objeto de facilitar el paso al siguiente. Los detalles restantes se determinan más tarde, cuando se conocen los requerimientos del negocio y se han pedido y efectuado cambios [9], [15], [29].

El ciclo de vida DSDM, define tres ciclos iterativos distintos, precedidos de dos actividades adicionales al ciclo de vida [30]:

Estudio de factibilidad: establece los requerimientos y restricciones básicas del negocio, asociados con la aplicación que se va a construir, para luego evaluar si la aplicación es un candidato viable para aplicarle el proceso.

Estudio del negocio: establece los requerimientos e información funcional que permitirán a la aplicación dar valor al negocio; asimismo, define la arquitectura básica de la aplicación e identifica los requerimientos para darle mantenimiento.

Iteración del modelo funcional: produce un conjunto de prototipos incrementales que demuestran al cliente la funcionalidad. El objetivo de este ciclo iterativo es recabar requerimientos adicionales por medio de la obtención de retroalimentación de los usuarios cuando practican con el prototipo.

Diseño e iteración de la construcción: revisa los prototipos construidos durante la iteración del modelo funcional a fin de garantizar que en cada iteración se ha hecho ingeniería en forma que permita dar valor operativo del negocio a los usuarios finales; la iteración del modelo funcional y el diseño e iteración de la construcción ocurren de manera concurrente.

Implementación: coloca el incremento más reciente

del software en el ambiente de operación.

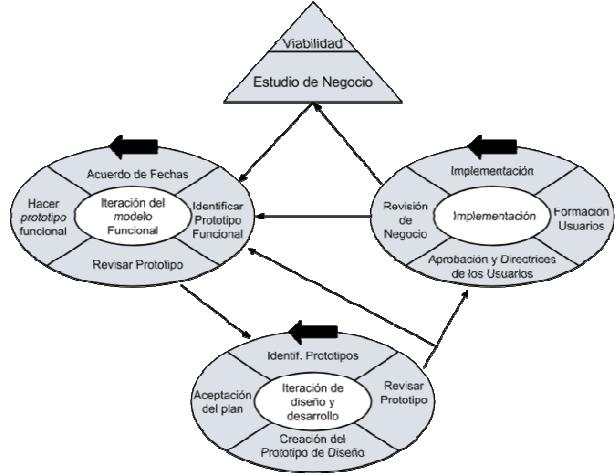


Fig. 6. Ciclo de vida DSDM [30].

III. ANÁLISIS COMPARATIVO ENTRE METODOLOGÍAS DE DESARROLLO ÁGILES

Aunque existen varias metodologías de desarrollo ágiles disponibles, cada una se aplica a un conjunto específico de proyectos. Un proyecto de desarrollo de software tiene varios factores asociados, como el tamaño del proyecto, la complejidad, el tiempo asignado, el presupuesto, etc. La selección de la metodología adecuada para el desarrollo de software depende de tales factores. Por lo tanto, un análisis comparativo de metodologías ágiles ayudará a decidir cuál se puede utilizar en una situación dada [31], [32].

A. Documentación:

Uno de los principios del desarrollo ágil es reducir la cantidad de tiempo y esfuerzo dedicado a la documentación. Pero la documentación que es importante no puede ser eliminada por completo [33], [34]:

- En metodologías como Scrum, XP y ASD la documentación es de menor importancia.
- En proyectos con FDD se requiere más documentación.
- DSDM requiere un nivel moderado de documentación que aún es menor que FDD.

B. Interacción con el cliente:

Las metodologías ágiles otorgan una importancia primordial a la comunicación frecuente con los usuarios finales. Aun así, el grado de implicación es diferente en cada metodología [35], [36]:

- XP y Scrum tienen una alta participación del cliente en el proceso de desarrollo.
- En ASD y DSDM, la participación del cliente o usuario final se puede ver durante el inicio y el final de la iteración.
- Mientras que FDD utiliza informes para comunicarse con los clientes.

C. Reuniones:

La comunicación es uno de los principios básicos establecidos en el manifiesto ágil. El éxito de las metodologías ágiles depende de la comunicación efectiva entre los miembros del equipo [37]:

- Las reuniones son de carácter informal y no se conserva documentación.
- Debido al uso de la técnica de programación en pares, el éxito de XP depende en gran medida de la comunicación.
- FDD y DSDM se basan en informes y documentación para la comunicación.
- Las reuniones cara a cara se utilizan para la comunicación en ASD.

D. Tamaño y complejidad del proyecto:

Cada metodología es adecuada para un tipo particular de proyecto:

- XP y ASD son usualmente preferidos para proyectos pequeños y menos complejos.
- XP es adecuado para proyectos donde hay un cambio constante en la especificación del producto.
- Scrum, FDD y DSDM se puede aplicar a cualquier tamaño de proyecto.

En la Tabla II y Tabla III se comparan las metodologías ágiles revisadas anteriormente con respecto a diferentes parámetros [36]:

TABLA II
COMPARACIÓN DE METODOLOGÍAS ÁGILES

Características	XP	SCRUM
Enfoque	Iterativo, incremental	Iterativo, incremental
Periodo del ciclo de iteración	1 – 6 semanas	2 – 4 semanas
Tamaño adecuado del proyecto y complejidad	Proyecto pequeño y sencillo	Para proyectos grandes y complejos
Involucramiento del usuario	Activamente involucrado	A través del propietario del producto
Documentación	Documentación básica	Documentación básica
Principales prácticas	Simplicidad, Programación en pares.	Reuniones de SCRUM
Desarrollo de características concurrentes	Possible	Possible

Nota: La tabla muestra una comparación entre las metodologías XP y SCRUM (Metodologías ágiles)

TABLA III
COMPARACIÓN DE METODOLOGÍAS ÁGILES

Características	DSDM	FDD	ASD
Enfoque	Iterativo	Iterativo	Iterativo, incremental

Periodo del ciclo de iteración	En 20% por ciento del tiempo total 80% de producto.	2 días - 2 semanas	4 – 8 semanas
Tamaño adecuado del proyecto y complejidad	Todo tipo de proyectos	Proyectos a gran escala	Proyectos más pequeños y sencillos
Involucramiento del usuario	A través de lanzamientos frecuentes	A través de informes	A través de lanzamientos frecuentes
Documentación	Más que XP y SCRUM	Más alto entre todos	Documentación básica
Principales prácticas	Time boxing, MoSCoW, Prototipado	Modelado de objetos, desarrollo por característica, uso del diagrama UML	Time boxing, Risk Driven, basado en características
Desarrollo de características concurrentes	Possible	Possible	Possible

Nota: La tabla muestra una comparación entre las metodologías DSDM, FDD y ASD (Metodologías ágiles)

En la Figura 7 se puede evidenciar los resultados del XII Informe Anual del Estado de las Metodologías Ágiles del 2017, en dónde se manifiesta que Scrum es la metodología ágil preferida de las organizaciones a nivel mundial:

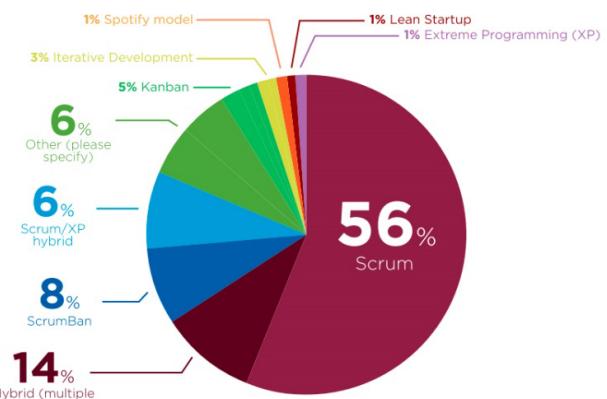


Fig. 7. Métodos y prácticas ágiles [12].

IV. CONCLUSIONES

- Las metodologías ágiles están ganando popularidad y ahora son preferidas a las metodologías de desarrollo de software tradicionales que tienen varios inconvenientes, como la incapacidad para hacer frente a los requisitos de usuario en constante cambio y exceder el tiempo y presupuesto asignados. Con los modelos de desarrollo de software tradicionales, los requisitos del producto deben especificarse claramente de antemano. Teniendo en cuenta el entorno empresarial actual, es importante que la metodología de

- desarrollo utilizada se adapte fácilmente a los requisitos cambiantes del usuario final.
- Las metodologías de desarrollo de software ágil manejan los requisitos cambiantes de los clientes a través de un enfoque iterativo e incremental. Tiene una iteración de desarrollo más corta con cada iteración o incremento seguido de pruebas y análisis de riesgos que resultan en un desarrollo y entrega más rápidos de un producto de calidad.
 - La selección de metodologías ágiles apropiadas es importante para maximizar la probabilidad de entrega de un producto de alta calidad que cumpla con los requisitos del usuario final. La comparación realizada en esta investigación se puede utilizar para decidir qué metodología se puede adaptar a un proyecto en particular.

REFERENCIAS

- [1] J. Mera and X. Quiñónez-Ku, "Análisis de los procesos de desarrollo de software en el departamento de TICs de la Pontificia Universidad Católica del Ecuador Sede Esmeraldas," *Pontificia Universidad Católica del Ecuador Sede Esmeraldas*, 2015.
- [2] O. Tinoco, P. Rosales, and J. Salas, "Criterios de selección de metodologías de desarrollo de software," *Ind. Data*, vol. 13, no. 2, p. 070, 2014.
- [3] Y. Amaya, "Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles," *Rev. Tecnol. | J. Technol.*, vol. 12 número, pp. 111–124, 2013.
- [4] D. A. Godoy, E. A. Belloni, H. Kotynski, H. dos Santos, and E. O. Sosa, "Simulado Proyectos de Desarrollo de Software Administrados con Scrum," *XVI Work. Investig. en Ciencias la Comput.*, pp. 485–489, 2014.
- [5] A. Navarro, J. Fernández, and J. Morales, "Revisión de metodologías ágiles para el desarrollo de software A review of agile methodologies for software development," *Univ. Icesi*, vol. 11 No. 2, pp. 30–39, 2013.
- [6] M. STOICA, M. MIRCEA, and B. GHILIC-MICU, "Software Development: Agile vs. Traditional," *Inform. Econ.*, vol. 17, no. 4/2013, pp. 64–76, 2013.
- [7] J. Canós, P. Letelier, and C. Penadés, "Métodologías Ágiles en el Desarrollo de Software," in *Métodologías Ágiles en el Desarrollo de Software*, Alicante: Universidad Politécnica de Valencia, 2003, pp. 1–8.
- [8] S. Overhage and S. Schlauderer, "Investigating the long-term acceptance of agile methodologies: An empirical study of developer perceptions in Scrum projects," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2012, pp. 5452–5461.
- [9] A. Moniruzzaman and S. Hossain, "Comparative Study on Agile software development methodologies," *arXiv Prepr. arXiv1307.3356*, vol. V, no. 3, pp. 37–56, 2013.
- [10] G. S. Matharu, A. Mishra, H. Singh, and P. Upadhyay, "Empirical Study of Agile Software Development Methodologies," *ACM SIGSOFT Softw. Eng. Notes*, vol. 40, no. 1, pp. 1–6, 2015.
- [11] A. Menzinsky, G. López, and J. Palacio, *Scrum Manager: Guía de formación*. Scrum Manager, 2016.
- [12] VersionOne, "12th Annual State of Agile Report," 2017.
- [13] A. Rosado, A. Quintero, and C. Meneses, "Desarrollo Ágil De Software Aplicando Programación Extrema," *Ingenio*, vol. 5, no. 1, pp. 2011–642, 2012.
- [14] P. Suardiyana, A. Yuliawati, and P. Mursanto, "Industrial Extreme Programming practice's implementation in rational unified process on agile development theme," in *2012 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2012, pp. 137–142.
- [15] R. Pressman, *Ingeniería del Software.- Roger.Pressman.6th.Ed.McGraw-Hill.pdf*.
- [16] Y. Yong and B. Zhou, "Evaluating extreme programming effect through system dynamics modeling," in *Proceedings - 2009 International Conference on Computational Intelligence and Software Engineering, CISE 2009*, 2009.
- [17] I. Sommerville, *INGENIERÍA DE SOFTWARE*, NOVENA EDI. México: PEARSON, 2011.
- [18] B. Xu, "Towards high quality software development with extreme programming methodology: Practices from real software projects," in *Proceedings - International Conference on Management and Service Science, MASS 2009*, 2009.
- [19] V. Temitayo, A. Badru, and N. Ajayi, "Adopting Scrum as an Agile Approach in Distributed Software Development: A Review of Literature," *Ieee*, vol. 1, no. 2, pp. 1–5, 2017.
- [20] K. Kaur, A. Jajoo, and Manisha, "Applying agile methodologies in industry projects: Benefits and challenges," in *Proceedings - 1st International Conference on Computing, Communication, Control and Automation, ICCUEA 2015*, 2015, pp. 832–836.
- [21] B. L. Romano and A. D. Da Silva, "Project management using the scrum agile method: A case study within a small enterprise," in *Proceedings - 12th International Conference on Information Technology: New Generations, ITNG 2015*, 2015, pp. 774–776.
- [22] A. Srivastava, S. Bhardwaj, and S. Saraswat, "SCRUM model for agile methodology," in *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017*, 2017, vol. 2017–Janua, pp. 864–869.
- [23] A. Ahmed, S. Ahmad, N. Ehsan, E. Mirza, and S. Z. Sarwar, "Agile software development: Impact on productivity and quality," in *2010 IEEE International Conference on Management of Innovation & Technology*, 2010, pp. 287–291.
- [24] M. Mahalakshmi and M. Sundararajan, "Tracking the student's performance in Web-based education using Scrum methodology," in *Proceedings of the International Conference on Computing and Communications Technologies, ICCCT 2015*, 2015, pp. 379–382.
- [25] M. García, "Estudio comparativo entre las metodologías ágiles y las metodologías tradicionales para la gestión de proyectos software," 2015.
- [26] A. A. Abdelaziz, Y. El-Tahir, and R. Osman, "Adaptive Software Development for developing safety critical software," in *Proceedings - 2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering, ICCNEEE 2015*, 2016, pp. 41–46.
- [27] A. F. Chowdhury and M. N. Huda, "Comparison between adaptive software development and feature driven development," in *Proceedings of 2011 International Conference on Computer Science and Network Technology, ICCSNT 2011*, 2011, vol. 1, pp. 363–367.
- [28] V. P. Doshi and V. Patil, "Competitor driven development: Hybrid of extreme programming and feature driven reuse development," in *1st International Conference on Emerging Trends in Engineering, Technology and Science, ICETETS 2016 - Proceedings*, 2016, pp. 1–6.
- [29] N. R. Mead, V. Viswanathan, and D. Padmanabhan, "Incorporating security requirements engineering into the dynamic systems development method," in *Proceedings - International Computer Software and Applications Conference*, 2008, pp. 949–954.
- [30] J. Jabeen *et al.*, "Incorporating artificial intelligence technique into DSDM," in *Asia-Pacific World Congress on Computer Science and Engineering, APWC on CSE 2014*, 2014.
- [31] Priyanka and P. Kantha, "A Comprehensive Study of Traditional and AGILE Software Development Methodologies," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 6, no. 11, pp. 128–138, 2016.
- [32] M. M. Kirmani, "Agile methods for mobile application development: A

- comparative analysis," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 5, pp. 1200–1205, 2017.
- [33] A. Mohammad, A. Khaled, A. Nidal, and T. Ahmed, "A Comparative Study of Agile Methods: XP versus SCRUM," *Int. J. Comput. Sci. Softw. Eng.*, vol. 4, no. 5, pp. 126–129, 2015.
- [34] R. P. Pawar, "A Comparative study of Agile Software Development Methodology and traditional waterfall model," *IOSR J. Comput. Eng.*, pp. 1–8, 2015.
- [35] W. C. D. S. Carvalho, P. F. Rosa, M. D. S. Soares, M. A. T. Da Cunha, L. C. Buiatte, and M. A. T. Da Cunha, "A comparative analysis of the agile and traditional software development processes productivity," in *Proceedings - International Conference of the Chilean Computer Science Society, SCCC*, 2012, pp. 74–82.
- [36] K. Hiwarkar, A. Doshi, R. Chinta, and R. Manjula, "Comparative Analysis of Agile Software Development Methodologies-A Review," *J. Eng. Res. Appl.*, vol. 6, no. 3, pp. 80–85, 2016.
- [37] A. B. M. Moniruzzaman and S. Akhter, "Comparative study on software development methodologies.," *Database Syst. J.*, vol. 5, no. 3, pp. 37–56, 2014.