Una revisión comparativa de la literatura acerca de metodologías tradicionales y modernas de desarrollo de software

A comparative review about traditional and modern software development methodologies

Sandra Milena Velásquez Restrepo [a], Juan David Vahos-Montoya [b], Marta Ester Gómez-Adasme [c], Ana Alexandra Pino – Martínez [d], Erika Juliett Restrepo-Zapata [e] & Sebastián Londoño-Marín [f]

[a] *Magíster en Ingeniería, SENA Centro de Servicios y Gestión Empresarial, Grupo de investigación GIGAT, Medellín, Colombia, smvelasquez@sena.edu.co*
[b] *Instructor, Centro de Servicios y Gestión Empresarial - CESGE, Grupo GIGAT, Servicio Nacional de Aprendizaje SENA, Medellín-Colombia, jvahosmo@sena.edu.co*
[c] *Instructora, Centro de Servicios y Gestión Empresarial CESGE, Servicio Nacional de Aprendizaje SENA - Grupo GIGAT, Medellín-Colombia, mega2808@misena.edu.co*
[d] *Investigadora Sennova, Centro de Servicios y Gestión Empresarial - CESGE, Grupo GIGAT, Servicio Nacional de Aprendizaje SENA, Medellín-Colombia, apinom@misena.edu.co*
[e] *Investigadora Sennova, Centro de Servicios y Gestión Empresarial - CESGE, Grupo GIGAT, Servicio Nacional de Aprendizaje SENA, Medellín-Colombia, ejrestrepo28@misena.edu.co*
[f] *Investigador Sennova, Centro de Servicios y Gestión Empresarial - CESGE, Grupo GIGAT, Servicio Nacional de Aprendizaje SENA, Medellín-Colombia, slondonom@misena.edu.co*

Recibido: abril 20, 2019. Aceptado: noviembre 30, 2019.

Resumen

Existen una gran variedad de metodologías de desarrollo de software, con diferencias significativas en cuanto al tipo de proceso empleado y productos obtenidos, cantidad de recursos involucrados, tiempos de desarrollo y estructura organizacional requerida. Considerando la diversidad de información disponible, este trabajo presenta una revisión enfocada a identificar las tendencias reportadas en la literatura sobre metodologías de desarrollo de software, incluyendo tanto las modernas como las tradicionales. Se aplicaron filtros de búsqueda para seleccionar estudios primarios y secundarios, encontrando que las metodologías de desarrollo tradicionales más reportadas son los de tipo cascada y modelos en espiral, mientras que entre las metodologías modernas las ágiles son las más reportadas, incluyendo programación extrema, scrum, desarrollo orientado a funcionalidades y las basadas en componentes. Se identificaron y compararon las principales ventajas y desventajas de las metodologías, buscando que este trabajo sirva como un punto de partida para los desarrolladores de software a nivel empresarial, educativo e investigativo.

Palabras Clave: *Metodologías de desarrollo de software, metodologías tradicionales, metodologías modernas, enfoque ágil..*

Abstract

There is a wide variety of software development methodologies, with significant differences in terms of processes, product, amount of resources involved, development times, and organizational structure required. This work presents a review focused on identifying the trends reported in the literature on software development methodologies, including both modern ones and those considered traditional. After an initial search in databases and repositories, filters for searching and selecting primary and secondary studies were applied. The research results showed that the most reported traditional development methodologies are cascade and spiral models. The most frequently reported modern methodologies are the agile methods, including extreme programming (XP), scrum, feature-driven development (FDD) and component-based software development (CBSD). The main advantages and disadvantages of the methodologies were identified and compared. This review can be used as an initial tool for software developers at a business, educational and research level.

Keywords: *Software development methodologies, traditional methodologies, modern methodologies, agile approach.*

Citar como:
Sandra Milena Velásquez Restrepo, Juan David Vahos-Montoya, Marta Ester Gómez-Adasme, Ana Alexandra Pino – Martínez, Erika Juliett Restrepo-Zapata & Sebastián Londoño-Marín. "Una revisión comparativa de la literatura acerca de metodologías tradicionales y modernas de desarrollo de software" Revista CINTEX, Vol. 24(2), pp. 13-23. 2019.

1 INTRODUCCIÓN

Es cada vez más común el uso de tecnologías que emplean software, como es el caso de los teléfonos celulares o los computadores personales, por citar solo dos ejemplos relevantes del uso actual del software [1]. La intensa dinámica de la industria del software exige buscar cómo atender las exigencias del mercado, motivando que se empleen diferentes metodologías o modelos de desarrollo de software [2], los cuales son estudiados en este trabajo.

Una metodología es una colección estructurada de procedimientos que ayudan a los desarrolladores de software en sus proyectos [3], ofreciendo una guía para la toma de decisiones, así como para planificarlo, gestionarlo, controlarlo y evaluarlo [4]. La elección de la metodología a emplear es clave durante el desarrollo de un software por sus implicaciones en lo referente a efectividad, eficiencia y desempeño del producto, costo y el tiempo de desarrollo, métodos de control de calidad y de pruebas, los cuales deben ajustarse a las particularidades de cada metodología [5].

Una metodología de desarrollo de software es similar a una receta de cocina. De la misma manera que una receta le enseña a uno cómo cocinar una comida, un método de desarrollo de software enseña cómo construir un producto de software. Las metodologías aumentan los esfuerzos para mejorar la calidad de los productos al mejorar los procesos que producen los productos. Se requiere un estándar o método formalmente definido para controlar los procesos de desarrollo. Las diferentes metodologías de desarrollo de software tienen fortalezas y debilidades. La elección de qué método usar realmente depende de los objetivos que una empresa quiera alcanzar.

Las metodologías de desarrollo de software se clasifican en tradicionales y modernas [4]. En este trabajo se describen las características, ventajas y desventajas más relevantes de ambos tipos de metodologías según los reportes de la literatura. Inicialmente se presenta el método empleado para realizar la revisión, luego se presentan los resultados obtenidos y su respectivo análisis, y finalmente se exponen las conclusiones, incluyendo recomendaciones para trabajos futuros.

2 METODOLOGÍA EMPLEADA PARA REALIZAR LA REVISIÓN

Inicialmente se consultaron publicaciones de los últimos cinco años en las bases de datos electrónicas especializadas, pero debido a la baja cantidad de trabajos sobre metodologías tradicionales se incluyeron también libros, actas de conferencia y tesis de grado ampliando la ventana a los últimos 10 años. El sistema de búsqueda usado consiste en una adaptación de la investigación realizada por Dybå y Togeir [6], siguiendo las palabras clave descritas en la Tabla 1.

Tabla 1. Estrategias de búsqueda empleadas en este trabajo.

Temática: metodologías tradicionales	**Temática: metodologías modernas**	**Temática: comparación de metodologías**
(traditional AND models AND software) OR (waterfall AND software) OR (“rapid application prototype” AND software) OR (spiral AND software) OR (“rational unified process” AND software) OR (V-shaped OR V AND software)	(agile OR modern OR “component-based” AND software) OR (extreme programming) OR (XP AND software) OR (scrum AND software) OR (crystal AND software AND (clear OR orange OR red OR blue)) OR (DSDM AND software) OR (FDD AND software) OR (component assembly OR component-based AND software) OR (ADS AND software)	(traditional) AND (agile OR modern) AND (models OR methodologies) AND (software)

Se revisaron títulos y resúmenes de los trabajos, excluyendo los que no se centraban en metodologías para el desarrollo de software modernas (ágiles y/o basadas en componentes) o tradicionales, o comparación de ambas. Se excluyeron trabajos que hacían un estudio general de la metodología pero que no definían o proponían un modelo para su aplicación en forma concreta, acogiendo las recomendaciones de Morales y Pardo [7]. Finalmente, se eligieron los estudios que realizaban los mejores aportes a cada temática. Es relevante mencionar que se tomaron estudios tanto primarios como secundarios, y que en el período de observación solo se encontró un estudio terciario, el cual estaba orientado a diversas temáticas relacionadas con metodologías ágiles [8].

3 METODOLOGÍAS TRADICIONALES DE DESARROLLO DE SOFTWARE

Estas metodologías imponen una disciplina de trabajo sobre el proceso de desarrollo del software, buscando conseguir un software más eficiente y predecible; este enfoque es considerado tradicional por ser el primero que se empleó para desarrollar software [9]. Exige que se preste gran atención a la planificación total de todo el trabajo a realizar y, una vez que está todo detallado, se inicia el desarrollo del producto [10]. En la Tabla 2 se presentan las ventajas y desventajas de las metodologías tradicionales más referenciadas en la literatura, y en las secciones subsiguientes, se describen las principales características de las metodologías tradicionales destacadas.

Tabla 2. Comparativo de metodologías tradicionales para el desarrollo de software

METODOLOGÍA	VENTAJAS	DESVENTAJAS
Cascada [1], [4], [9], [13]	-No se mezclan las fases de desarrollo -La planificación es sencilla -La calidad del producto es alta -Es un modelo sencillo y de fácil aprendizaje	- Definir los requisitos al comienzo de un proyecto, incorporar nuevos requerimientos, e integrar la gestión del riesgo es difícil - El tiempo de entrega es mayor que el de otras metodologías, y la calidad del producto se afecta al cambiar el orden de las fases - Las interacciones con los usuarios son más costosas que con otras metodologías - No se adapta bien a proyectos grandes, ya que los defectos se identifican al final del proyecto
En V [11], [14], [18]	- Se realizan pruebas en etapas intermedias del ciclo de vida, por lo cual tiene mayor probabilidad de éxito que el modelo de cascada. - Involucra al usuario en las pruebas. -Funciona bien para proyectos pequeños.	- Es difícil que el cliente exponga explícitamente todos los requisitos -Las pruebas pueden ser costosas. - Muy rígido y poco flexible, lo cual puede ser costoso. - No se producen prototipos iniciales del software. - No proporciona una ruta clara para los problemas encontrados durante las fases de prueba.
Espiral [3], [4], [11], [19]	- Permite aplicar el enfoque de construcción de los prototipos en cualquier etapa. - Más flexible que cascada y en V.	- Más complejo, implicando mayores tiempos de desarrollo y costos - Falta de orientación explícita del proceso para determinar objetivos y restricciones. - Más útil para grandes y costosos proyectos, pues busca la reducción de riesgos
Proceso racional unificado (RUP) [4], [11], [15]–[17]	- Es la metodología más utilizada para sistemas orientados a objetos	- El grado de complejidad no es adecuado para proyectos pequeños
Desarrollo de aplicaciones rápidas RAD [11]	-Basada en co-creación entre desarrollador y usuario	-No es recomendable si la participación del usuario no es activa.

3.1 Cascada

También llamado ciclo de vida clásico o lineal; plantea un enfoque sistemático y secuencial para el desarrollo del software [11]. Es la metodología tradicional más empleada, y refleja fielmente los principios de las metodologías convencionales [11], [12]. Los proyectos se tratan de forma predictiva, midiendo el progreso en términos de artefactos entregados, especificación de los requisitos, documentos de diseño, planes de pruebas y revisiones de código [13].

3.2 Modelo en V

Se considera una extensión del modelo de cascada, pero en lugar de avanzar en forma lineal, los pasos del proceso se doblan después de la fase de codificación, adquiriendo forma de "V" para efectuar procesos de validación y verificación [1], [11].

3.3 Modelo en espiral

La dimensión radial del modelo representa los costos acumulados, y el ángulo representa el progreso realizado al completar cada ciclo. Como se esquematiza en la Figura 1, cada bucle de la espiral representa una fase [11], que se completa con una revisión de diseñadores y programadores [4].

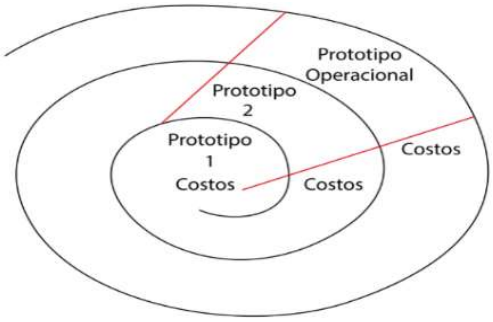

Figure 1. Modelo espiral. Fuente: adaptado de [11]

3.4 Proceso racional unificado (RUP)

El RUP se organiza en torno a fases y flujos de trabajo, empleando un enfoque iterativo para sistemas orientados a objetos [11], [15]. La metodología no tiene etapas establecidas rígidamente, pues se adapta a las necesidades de cada organización, dándole especial importancia a la construcción ágil de prototipos [16], [17].

3.5 Modelo de desarrollo rápido de aplicaciones (RAD)

En la metodología se crea un prototipo rápido y se le entrega al usuario para que lo evalúe, y así obtener retroalimentación. Luego, se refina el prototipo con base en los comentarios de los usuarios.

4 METODOLOGÍAS MODERNAS DE DESARROLLO DE SOFTWARE

Estas metodologías se centran en la integración de componentes o en la capacidad de adaptación a los requerimientos que surgen en el desarrollo, de allí su nombre metodologías ágiles. En la Tabla 3 se presenta un compendio de las ventajas y desventajas de las metodologías modernas más referenciadas en la literatura, y en las secciones subsiguientes, se describen las principales características de las metodologías modernas destacadas.

Tabla 3. Comparativo de metodologías modernas para el desarrollo de software

Metodología	Ventajas	Desventajas
Metodología basada en componentes [11], [20], [21]	- Como los componentes se reutilizan, pueden ser más confiables que el software desarrollado desde cero, ya que se probaron en una gran variedad de condiciones. - Útil para desarrollar softwares complejos.	-La calidad depende de la manera como se realiza la integración. El rendimiento varía pues las pruebas a componentes pueden ser limitadas -Como involucra muchas personas, es difícil saber qué desarrolla cada una.
Scrum [7], [26], [34], [36], [43], [46]	- Aplicable a proyectos con cambios de requisitos. -Ha sido adoptada en prácticas orientadas a CMMI o PSP/TSP.	- Mayor riesgo de generar estrés en el equipo de trabajo debido a los continuos Sprint.
Programación extrema (XP) [3], [6], [11], [14], [23], [29], [35], [43], [44]	- Favorece la productividad. - Adecuada para proyectos con requisitos imprecisos y cambiantes, donde hay un alto riesgo técnico. - Promueve las relaciones interpersonales y el aprendizaje de los desarrolladores.	- Los desarrolladores son los mismos que aprueban los códigos de programación. - Es difícil de introducir en organizaciones grandes. - No es adecuada para grandes proyectos ni grupos grandes de trabajo
Crystal [18], [19], [36]	- Posibilidad de seleccionar el método más adecuado según el tamaño y la criticidad del proyecto.	-Si el proyecto es muy complejo, un error no detectado puede ser crítico. - Aumentar la envergadura del proyecto añade costo.
Desarrollo de sistemas dinámicos (DSDM) [5], [18], [32], [38]	- Evolucionan en el tiempo adaptándose a la necesidad. - Ha demostrado ser útil para pequeñas y medianas empresas.	- Mayor riesgo de generar estrés en el equipo de trabajo pues la metodología exige entregas a tiempo. - La calidad del resultado depende de las bases sólidas al inicio del desarrollo.
Desarrollo adaptativo de software (ASD) [3], [18], [25]	- La atención se centra más en los resultados y su calidad que en las tareas o el proceso utilizado para producir el resultado. - No impone el uso de equipos co-ubicados como la mayoría de las demás metodologías ágiles.	- Construir sistemas complejos requiere un amplio conocimiento de muchas tecnologías, requiriendo equipos especializados. - No se ha presentado una investigación significativa sobre ASD, posiblemente porque el método deja mucho espacio para la adaptación.
Desarrollo orientado a funcionalidades (FDD) [6], [11].	- Ideada para trabajar con otras metodologías de desarrollo de software. - No utiliza un modelo de proceso específico.	- La planeación y el diseño deben realizarse por adelantado, lo cual imprime cierta rigidez de las metodologías tradicionales.
Desarrollo esbelto (LSD) [28], [39]	- Elimina tareas que no aportan valor	- Requiere personal abierto al aprendizaje continuo.
Proceso unificado ágil (AUP) [24]	Fácil de aprender y modificar, Adaptable (permite realizar cambios, aprender y modificar)	- Exige gran disponibilidad del cliente para trabajar con los desarrolladores.
Proceso unificado abierto (OpenUP) [39]	- Es utilizada como base, y adaptar su contenido a la metodología que mejor se ajuste a los requerimientos del proyecto.	- No orienta sobre el tamaño del equipo requerido, cumplimiento, seguridad u orientación tecnológica.

4.1 Metodologías de desarrollo de software basado en componentes (CBSD)

Integra componentes comerciales que se puedan emplear para construir un sistema de software [20], facilitando el desarrollo de aplicaciones complejas y la implementación en diferentes plataformas [21].

4.2 Metodologías ágiles

Los usuarios potenciales se involucran en las pruebas al producto desde etapas tempranas del proyecto [22], buscando resultados en menor tiempo sin disminuir su calidad [23]. El cliente indica lo que espera a medida que recibe entregas tempranas y constantes, lo cual requiere su compromiso con los desarrolladores [24]. Los principios de estas metodologías se agrupan en el "Manifesto for Agile Software Development" [16], siendo ellos [25], [26]: satisfacción del cliente, adaptación al cambio, entregables de software, trabajo en equipo, motivación, diálogo, software funcional, desarrollo sostenible, atención continua, simplicidad, organización y efectividad. A continuación, se describen las características de las metodologías ágiles más referenciadas en la literatura.

4.2.1 Metodología Scrum

Es la metodología ágil más utilizada [27], diseñada para lograr la colaboración eficaz de los diferentes equipos relacionados con el proyecto [2[1]8]. Se realizan sprints o entregas iterativas del producto al cliente para que lo pruebe y realice observaciones [29]. Deben efectuarse reuniones diarias de máximo 15 minutos con el equipo de trabajo para coordinar el proyecto debidamente [29]–[31], y asignar roles claros a los miembros del equipo desarrollador [23], [29], [32]. En la revisión bibliográfica realizada por Vallon et al. [33], ocho de las diez metodologías ágiles más usadas en el mundo en el período 2010–2016 se basan en el modelo scrum.

4.2.2 Programación extrema (XP)

La metodología se basa en cinco valores: simplicidad, comunicación, respeto y coraje para alcanzar retroalimentación rápida, simplicidad, cambio

4.2.3 Proceso unificado ágil (AUP)

El AUP aplica técnicas ágiles incluyendo desarrollo dirigido por pruebas (TDD por sus siglas en inglés), modelado ágil, gestión ágil de cambios, y refactorización de base de datos para mejorar la productividad. El ciclo de vida del AUP tiene cuatro fases: concepción, donde se define el alcance y las arquitecturas para el software, elaboración en donde el equipo de desarrollo comprende los requisitos y valida la arquitectura del sistema, construcción en la cual el sistema es desarrollado y finalmente se despliega [40].

4.2.4 Proceso unificado abierto (OpenUP)

Esta metodología se caracteriza por el desarrollo incremental, uso de casos de uso y escenarios, manejo de riesgos, y diseño basado en la arquitectura [41].

5 COMPARATIVO ENTRE METODOLOGÍAS TRADICIONALES Y MODERNAS

Las metodologías tradicionales se basan en planes en los que el trabajo comienza con la obtención y documentación de un conjunto completo de requisitos, seguido de una selección de arquitectura y una etapa de alto nivel, desarrollo de diseño e inspección. Sin embargo, algunos profesionales encontraron este proceso demasiado centrado en el software, con un desarrollo frustrante y planteando dificultades aun cuando los cambios en el diseño son relativamente pequeños. Como una respuesta a estas problemáticas, se desarrollaron nuevas metodologías y prácticas basadas en mejoras iterativas, lo que llevó al desarrollo de las metodologías ágiles. El nombre "ágil" surgió en 2001, cuando diecisiete metodólogos de procesos se reunieron para discutir las tendencias futuras en el desarrollo de software. Se dieron cuenta de que sus métodos tenían muchas características en común, por lo que decidieron nombrar estos procesos ágiles, lo que significa que son a la vez ligeros y suficientes.

La metodología ágil tiende a exponer la disfunción organizacional. A diferencia de los métodos tradicionales, las metodologías ágiles utilizan iteraciones en lugar de fases, empleando ciclos iterativos cortos, lanzamientos pequeños / cortos, diseño simple, favoreciendo la integración continua y confiando en el conocimiento implícito dentro de un equipo más que en la recopilación de documentación.

La diferencia clave entre las metodologías tradicionales y las modernas es el factor de adaptabilidad.

En una metodología ágil, si se requiere algún cambio importante, el equipo no detiene su flujo de trabajo; más bien determina cómo manejar mejor los cambios que ocurren durante todo el proyecto. El proceso de verificación en el método ágil ocurre mucho antes en el proceso de desarrollo. Por otro lado, los métodos tradicionales manuales congelan los requisitos del producto y no permiten el cambio, pues implementan un proceso predictivo que se basa en definir y documentar un conjunto estable de requisitos al inicio de un proyecto.

En la Tabla 4 se presenta una comparación entre las metodologías abordadas en este trabajo, de acuerdo con ciertas características o aspectos definidos en [46], [47].

Tabla 4. Comparativo de metodologías agiles y tradicionales

ASPECTO	**METODOLOGÍA**	
	TRADICIONAL	**MODERNA**
Resistencia al cambio durante el proyecto	Alta [1], [3], [14]	Baja [1], [13], [14]
Políticas/normas/ control	Numerosos [3]	Pocos [17]
Tipo de contrato con el cliente	Fijo para el proyecto [3], [15],	Flexible [3], [15], [14]
Tamaño de los equipos de trabajo	Grandes [15],	Pequeños [3], [13]
Roles de los integrantes	Específicos [15]	Flexibles [3], [4]
Énfasis en la arquitectura del software	Muy alto [3], [15], [17],	Bajo [15], [22]
Enfoque del proyecto	Se concibe como un gran proyecto [15]	Se subdivide en pequeños proyectos [13]
Cantidad de documentación	Extensa [3], [13]	Poca [17]
Entrega del producto	Al final del proyecto [15]	Entregas constantes de prototipos [13], [15], [17],
Frecuencia de comunicación y forma de interacción con el cliente	Poca / Mediante reuniones [3],	Constante / Forma parte del equipo de desarrollo [22], [25]
Centro de la metodología	Procesos: roles, actividades, artefactos [14]	Las personas y el trabajo en equipo [17]
Costo y envergadura de los proyectos	Altos [22]	Bajos [15], [22]
Momento en que se define la arquitectura del software	Etapa temprana del proyecto [15],	Se define y mejora durante el proyecto [22]
Enfoque hacia el aseguramiento de la calidad	Alto [11], [15]	No se efectúa control de calidad tradicional [1]

Si bien la calidad es más complicada de conseguir usando metodologías modernas, con éstas se puede lograr mayor satisfacción del cliente [46], [47] por la mejor entrega presupuestaria y el grado de oportunidad logrado [47]. Otro aspecto que hace atractivas las metodologías modernas es que son más fáciles de acoger por personas que no están acostumbradas a seguir procesos [48]. Una tendencia reportada es combinar metodologías, resaltando las experiencias con programación extrema y scrum [23], [38] así como versiones híbridas según la necesidad específica [49].

5.1 Ejemplos de aplicaciones referenciadas en literatura

En la Tabla 5 se presenta un comparativo de aplicaciones desarrolladas con metodologías tradicionales y modernas,

Tabla 5. Comparativo de metodologías agiles y tradicionales

Nombre aplicación	**Metodología**
Sistema de gestión meteorológica [50]	Tradicional tipo RUP
sistema de control de almacén para la administración de alimentos en el Centro Juvenil El Tambo [53]	Tradicional tipo RUP
Aplicación de Scrum en el desarrollo de software en TeamSoft S.A.C.” [54]	Moderna tipo SCRUM
Sistema de código de barras para control de asistencia en una junta administradora [52]	Moderna tipo XP
Sistema de gestión de proyectos de desarrollo de sistemas informáticos para la Empresa Grupo “SAM” E.I.R.L [51]	Tradicional tipo RUP
Plataforma de gestión Tecnoparque Colombia [55]	Moderna tipo MVC
SAM - Herramienta de software para la gestión del mantenimiento de infraestructura en el SENA regional Antioquia [56]	Moderna tipo SCRUM
SIMA - Software para la gestión del mantenimiento en los laboratorios de la I.U. Pascual Bravo [57]	Tradicional tipo RAD

6 CONCLUSIONES Y RECOMENDACIONES

Existe una amplia diversidad de metodologías de desarrollo de software, pero a pesar de ser un campo de trabajo sumamente dinámico en la actualidad, en bases de datos especializadas hay una cantidad relativamente pequeña de estudios secundarios y primarios sobre la temática, así como solo un estudio terciario sobre aspectos muy generales de las metodologías ágiles. Se encontraron pocos estudios sobre metodologías tradicionales, lo cual confirma lo reportado en algunos estudios sobre la necesidad de motivar a la industria del software para que incremente la publicación de trabajos académicos sobre logros alcanzados en lo relacionado con metodologías de desarrollo de software.

Dado que existe abundante literatura gris sobre la temática, es pertinente establecer criterios para realizar estudios secundarios como el presentado en este trabajo, para que se recopile información documental que puede ser valiosa para los desarrolladores a la hora de seleccionar una metodología de software apropiada para un proyecto específico.

Las metodologías de desarrollo tradicionales más reportadas en la literatura son las de tipo cascada, iteración y modelos en espiral, mientras que las metodologías ágiles más frecuentes incluyen programación extrema, scrum, desarrollo orientado a funcionalidades y las basadas en componentes. En este trabajo se presentaron las principales características reportadas en la literatura para metodologías tradicionales y ágiles, sus ventajas y desventajas, estableciendo las diferencias más significativas entre ellas para que se tengan criterios de selección de metodologías de desarrollo de software según las particularidades de los proyectos.

REFERENCIAS

[1] H. D. Ortiz Alzate, L. G. Muñoz Marín, J. Cardeño Espinosa, y N. C. Alzate Osorno, «Impacto del uso de objetos interactivos de aprendizaje en la apropiación de conocimiento y su contribución en el desarrollo de competencias matemáticas: un resultado de experiencia de investigación», Rev. CINTEX, vol. 21, n.o 1, pp. 71-88, jun. 2016.
[2] V. Tiwari, «Software Engineering Issues in Development Models of Open Source Software», International Journal of Computer Science and Technology, vol. 2, n.o 2, pp. 38-44, 2011.
[3] A. Navarro Cadavid, J. D. Fernández Martínez, y J. Morales Vélez, «Revisión de metodologías ágiles para el desarrollo de software», Prospectiva, vol. 11, n.o 2, pp. 30–39, 2013.
[4] Y. D. Amaya Balaguera, «Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles», Revista de Tecnología, vol. 12, n.o 2, pp. 111–124, 2013.
[5] A. Peralta y F. P. Romero, «Toma de Decisiones a partir de Conocimiento Extraído tras el Análisis de Comportamientos Previos. Aplicación Práctica para la Dirección de Proyectos de Desarrollo de Software», Rev. CINTEX, vol. 20, n.o 2, pp. 97-111, dic. 2015.
[6] T. Dybå y D. Torgeir, «Empirical studies of agile software development: a systematic review», Information and software technology, vol. 50, pp. 833-859, 2008.
[7] J. J. Morales Arias y C. J. Pardo Calvache, «Revisión sistemática de la integración de modelos de desarrollo de software dirigido por modelos y metodologías ágiles», Informador Técnico, vol. 80, n.o 1, pp. 87-99, 2016.
[8] R. Hoda, N. Salleh, J. Grundy, y H. M. Tee, «Systematic literature reviews in agile software development: A tertiary study», Inf. Softw. Technol., vol. 85, pp. 60-70, may 2017.
[9] J. A. Mera Paz, «Análisis del proceso de pruebas de calidad de software», Ingeniería Solidaria, vol. 12, n.o 20, pp. 163-176, 2016.
[10] Ó. Tinoco Gómez, P. P. Rosales López, y J. Salas Bacalla, «Criterios de selección de metodologías de desarrollo de software», Industrial Data, vol. 13, n.o 2, 2010.
[11] G. Kumar y P. Kumar Bhatia, «Comparative analysis of software engineering models from traditional to modern methodologies», presentado en Fourth International Conference on Advanced Computing & Communication Technologies, Rohtak, Haryana, India, 2014, pp. 189-196.
[12] J. S. Restrepo Ángel, «Guía de buenas prácticas aplicable a la metodología de desarrollo ágil SCRUM para fortalecer la seguridad de la información», Trabajo de grado, Institución Universitaria Politécnico Grancolombiano, Bogotá, Colombia, 2017.
[13] E. Parra Castrillón, «Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje -MESOVA», Revista Virtual Universidad Católica del Norte, vol. 34, pp. 113-137, 2011.
[14] A. Orjuela Duarte y M. Rojas C., «Las metodologías de desarrollo ágil como una oportunidad para la ingeniería del software educativo», Revista Avances en Sistemas e Informática, vol. 5, n.o 2, 2008.
[15] R. G. Figueroa, C. J. Solís, y A. A. Cabrera, «Metodologías tradicionales vs. metodologías ágiles». Universidad Técnica Particular de Loja - Escuela de Ciencias en Computación, 2008.
[16] E. R. Casas-Huamanta, E. Linares-Fernández, y Y. Acuña-Huamán, «Metodologías ágiles para el desarrollo de aplicaciones móviles», presentado en IV Congreso Nacional de Investigación - CONACIN, Tarapoto, Perú, 2014.
[17] O. A. Pérez A., «Cuatro enfoques metodológicos para el desarrollo de software RUP-MSF-XP-SCRUM», Inventum, vol. 10, pp. 64-78, 2011.
[18] P. Abrahamsson, S. Outi, J. Ronkainen, y J. Warsta, «Agile software development methods: review and analysis». 2017.
[19] D. E. Strode, S. L. Huff, y A. Tretiakov, «The impact of organizational culture on agile method use», en Proceedings of the 42nd Hawaii International Conference on System Sciences - 2009, Hawaii, 2009, pp. 1-9.
[20] M. M. Amine y M. Ahmed-Nacer, «An agile methodology for implementing knowledge management systems : a case study in component-based software engineering», International Journal of Software Engineering and Its Applications, vol. 5, n.o 4, pp. 159-170, 2011.
[21] M. Yahlali y A. Chouarfia, «Towards a software component assembly evaluation», IET Softw., vol. 9, n.o 1, pp. 1-6, feb. 2015.
[22] M. E. Navarro, M. P. Moreno, J. Aranda, L. Parra, J. R. Rueda, y J. C. Pantano, «Selección de metodologías ágiles e integración de arquitecturas de software en el desarrollo de sistemas de información», presentado en XIX Workshop de Investigadores en Ciencias de la Computación - WICC 2017, Buenos Aires, Argentina, 2017, pp. 632–636.
[23] E. Ávila Domenech y A. Meneses Abad, «Comparative evaluation of Delfdroid whit XP and scrum using the 4- DAT», Revista Cubana de Ciencias Informáticas, vol. 7, n.o 1, pp. 16-23, 2013.

[24] T. Pozo, C. Aucancela, C. Hinojosa, y A. Abdelrahaman, «Sistema Web de Asignación de Aulas de los Laboratorios de Computación de la ESPE, Aplicando la Metodología Agile Unified Process (AUP), utilizando el Framework Junit», GEEKs-DECC Report -Tendencias en Computación, vol. 3, n.o 1, pp. 6-14, 2011.
[25] L. F. Betancur Cartagena, «Propuesta estratégica de prácticas seguras para el desarrollo de software con metodologías ágiles», Tesis de Maestría, Universidad Nacional de Colombia - Sede Medellín, Medellín, Colombia, 2016.
[26] A. Plaza Cordero, M. Arcos Argudo, y R. Bojorque Chasi, «Scrum en la educación: Caso de estudio como método de trabajo." INCISCOS 2016. 2017», en Proceedings Book, Quito - Ecuador, 2016, pp. 187-191.
[27] B. Corona, M. Muñoz, J. Miramontes, J. A. Calvo-Manzan, y T. San Feliu, «Estado de arte sobre métodos de evaluación de metodologías ágiles en las pymes», ReCIBE - Revista Electrónica de Computación, Informática, Biomédica y Electrónica, vol. 5, n.o 1, 2016.
[28] P. Nidagundi y L. Novickis, «Introducing Lean Canvas Model Adaptation in the Scrum Software Testing», Procedia Comput. Sci., vol. 104, pp. 97-103, 2017.
[29] P. C. Marecos Brizuela, «Revisión sistemática sobre metodologías agiles en empresas de software», Revista de la Facultad de Ciencias Aplicadas, Universidad Nacional de Pilar, vol. 1, pp. 54-72, 2017.
[30] P. Rodríguez, K. Mikkonen, P. Kuvaja, M. Oivo, y J. Garbajosa, «Building lean thinking in a telecom software development organization: strengths and challenges», 2013, p. 98.
[31] W. G. Barrios, M. V. Godoy Guglielmone, M. G. Fernández, S. I. Mariño, F. M. Ferreira, y C. T. Zarrabeitia, «SCRUM: application experience in a software development PyME in the NEA», Journal of Computer Science & Technology, vol. 12, n.o 3, pp. 110-115, 2012.
[32] A. Babativa, P. Briceño, C. Nieto, y O. Salazar, «Desarrollo ágil de una aplicación para dispositivos móviles. Caso de estudio: taxímetro móvil», Ingeniería, vol. 21, n.o 3, pp. 260-275, 2016.
[33] R. Vallon, B. J. da Silva Estácio, R. Prikladnicki, y T. Grechenig, «Systematic literature review on agile practices in global software development», Inf. Softw. Technol., dic. 2017.
[34] J. D. Yepes González, «AgileFM: Modelo de desarrollo ágil formal basado en la ISO/IEC 29110 para las micro, pequeñas y medianas empresas», Tesis de Maestría, Universidad EAFIT, Medellín, Colombia, 2016.
[35] L. E. Gimson Saravia, «Metodologías ágiles y desarrollo basado en conocimiento», Trabajo de especializacion, Universidad Nacional de La Plata, Argentina, La Plata, Buenos Aires, Argentina, 2012.
[36] L. M. Montoya Suarez, J. M. Sepúlveda Castaño, y L. M. Jiménez Ramos, «Análisis comparativo de las metodologías ágiles en el desarrollo de software aplicadas en Colombia», en Gestión del Talento Humano: Enfoques y Modelos, Corporación Centro Internacional de Marketing Territorial para la Educación y el Desarrollo - CIMTED, 2016, pp. 450-464.
[37] S. C. Gaona Bautista, «Modelo de procesos y gobernabilidad para el desarrollo ágil de software en TigoUNE», Tesis de Maestría en Administración, Universidad Nacional de Colombia - Sede Medellín, Medellín, Colombia, 2017.
[38] L. K. Roses, A. Windmöller, y E. A. do Carmo, «Favorability conditions in the adoption of agile method practices for software development in a public banking», J. Inf. Syst. Technol. Manag., vol. 13, n.o 3, dic. 2016.
[39] J. Pernstål, R. Feldt, y T. Gorschek, «The lean gap: A review of lean approaches to large-scale software systems development», J. Syst. Softw., vol. 86, n.o 11, pp. 2797-2821, nov. 2013.
[40] C. Edeki, «Agile Unified Process», Int. J. Comput. Sci. Mob. Appl., vol. 1, pp. 13-17, 2013.
[41] J. A. Britto Montoya, «Comparación de metodologías ágiles y procesos de desarrollo de software mediante un instrumento basado en CMMI», Scientia et technica, vol. 21, n.o 2, pp. 150-155, 2016.
[42] G. S. Matharu, A. Mishra, H. Singh, y P. Upadhyay, «Empirical Study of Agile Software Development Methodologies: A Comparative Analysis», ACM SIGSOFT Softw. Eng. Notes, vol. 40, n.o 1, pp. 1-6, feb. 2015.
[43] J. Dolado y D. Rodríguez, «Utilidad de los procesos ágiles en el desarrollo de software.», Novática - Revista de Asociación de Técnicos de Informática, vol. 209, pp. 73-74, 2011.
[44] J. H. Canós, M. C. Penadés, y P. Letelier, «Métodologías Ágiles en el Desarrollo de Software». DSIC - Univ. Politécnica Valencia, 2012.
[45] P. Blanco, J. Camarero, A. Fumero, A. Werterski, y P. Rodríguez, «Metodología de desarrollo ágil para sistemas móviles. Introducción al desarrollo con Android y el iPhone». Doctorado en Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, 2009.
[46] G. Villasana y R. Castello, «An agile software quality framework lacking», 2014, pp. 1-4.
[47] P. Jain, L. Ahuja, y A. Sharma, «Current state of the research in agile quality development», presentado en 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2016.
[48] L. Garcés y L. M. Egas, «Evolución de las metodologías de desarrollo de la ingeniería de software en el proceso la ingeniería de sistemas software», Rev. Científica Tecnológica UPSE, vol. 1, n.o 3, oct. 2015.
[49] J. A. Ruiz-Vanoye et al., Metodologías de Desarrollo de Software, Primera Edición. Editorial Académica Dragón Azteca, 2017.

[50] J. W. Cárdenas Rojas, "Implementación de un sistema de gestión de información de las estaciones meteorológicas, en la cuenca del Rio Cachi, Región Ayacucho." (2017).
[51] C. Sánchez, and M. Charly. "Implementación de un sistema de gestión de proyectos de desarrollo de sistemas informáticos para la Empresa Grupo “SAM” EIRL." (2017).
[52] I. D. Rivera Meza. "Desarrollo e implementación de un sistema de código de barras con la metodología XP para optimizar el control de asistencia en la junta administradora de Servicios de Saneamiento Quilcas." (2017).
[53] B. J. Salamán Herrera. "Implementación de un sistema de control de almacén para la administración de alimentos en el Centro Juvenil El Tambo." (2017).
[54] L. A. Villalva Castañeda. "Aplicación de Scrum en el desarrollo de software en TeamSoft SAC." (2017).
[55] J. J. Castro-Maldonado, J. A. Londoño-Gallego, S. Londoño-Marin, y J. A. Patiño-Murillo, «Implementation of a technological, information, and communication tool for project management in the network of Tecnoparque, Colombia», J. Phys. Conf. Ser., vol. 1418, p. 012014, dic. 2019, doi: 10.1088/1742-6596/1418/1/012014.
[56] J. D. Vahos, A. A. Pino, y J. J. Castro Maldonado, «Desarrollo de una herramienta de software para la gestión del mantenimiento de infraestructura en el SENA regional Antioquia», Rev. CINTEX, vol. 24, n.o 1, pp. 13-19, dic. 2019, doi: 10.33131/24222208.331.
[57] M. I. Ardila Marín, W. Orozco Murillo, J. Galeano Echeverri, y A. M. Medina Escobar, «Desarrollo de software para la gestión del mantenimiento en los laboratorios de la I.U. Pascual Bravo», Rev. CINTEX, vol. 23, n.o 1, pp. 43-50, oct. 2018.