Metodología tradicional vs ágil para la gestión de proyectos de software

Melissa Alvarez Martell1
melalvareez@gmail.com
Mariana Marcelino-Aranda1*
mmarcelino@ipn.mx
Alberto Macías Alcibar1
maalberto.rt@gmail.com
José Carlos Novoa Sandoval1
josecarlosnovoa@hotmail.com

Instituto Politécnico Nacional
Unidad Profesional Interdisciplinaria de Ingeniería
y Ciencias Sociales y Administrativas

Boletín No. 83
1o. de marzo de 2021

RESUMEN

Tras un acelerado crecimiento del internet y la tecnología desde 1970, el campo del desarrollo de sistemas de software se enfrentó a la necesidad constante de modificar los requerimientos del proyecto y del involucramiento de los usuarios del sistema, lo que derivó a transitar de metodologías tradicionales hacia las ágiles. El objetivo de este artículo, a partir de una revisión de la literatura, fue identificar las características y necesidades del desarrollo de software que dieron lugar a dicho cambio y los factores que permiten la viabilidad de éstas en la actual industria. Las metodologías ágiles representan beneficios para miembros del equipo de desarrollo y clientes como son reducción de costos, capacidad de predicción conforme se entregan las iteraciones, trabajo colaborativo y mayor satisfacción y motivación cliente–equipo. A pesar de las diversas ventajas potenciales que brindan los SDM los profesionales presentan cierta resistencia a la adopción de dichas metodologías (ágiles o tradicionales). Actualmente, existen metodologías hibridas entre metodologías tradicionales-ágiles y ágiles-ágiles, que pretenden combinar los mejores factores de las ya existentes.

s

INTRODUCCIÓN

La interacción persona-computadora está presente desde 1936 con la primera computadora “Z1 de Zonrad Zuse”. La computadora ha tenido una constante evolución en aspectos del hardware (elementos físicos que constituyen el sistema informático) y software (lógica que contiene el sistema informático para funcionar). El software establece el funcionamiento y la lógica que posibilita que el usuario

interactúe, a través del hardware, con el sistema informático o dispositivo. En las últimas dos décadas su desarrollo se basó en las notaciones de modelado y del desarrollo de herramientas que apoyan a los programadores de software. Por ejemplo, java, PHP, HTML, por mencionar algunas de las más conocidas. Sin embargo, los resultados no eran satisfactorios, a falta de una metodología que proviera de las directivas para su aplicación (Canós y Penadés, 2003). En aras de conocer tales metodologías, el objetivo de este artículo es identificar las características y necesidades del desarrollo de software que dieron lugar a la evolución de las metodologías tradicionales hacia las ágiles, así como de los factores que permiten la viabilidad de éstas en la actual industria.

METODOLOGIA

Este trabajo contiene información obtenida de bases de datos (Web of Science) e índices de revistas (Scielo, Scopus) y referencias bibliográficas (libros y tesis) que ayudaron a conocer y analizar los antecedentes de la computación y de las metodologías tradicionales y ágiles. La información recopilada se dividió en dos fases. En la primera, se llevó a cabo una búsqueda general de los antecedentes, bases conceptuales y ejemplos de las metodologías ágiles y tradicionales para la gestión de proyectos de software (SCRUM, XP, metodología de cascada, espiral, entre otros). Se utilizó un criterio de exclusión, coincidencia con el tema de investigación, del título y el resumen de los artículos, libros y tesis. Se obtuvo un total de 20 artículos.

La segunda fase implicó una búsqueda y análisis de los sinónimos y palabras claves que se identificaron y recuperaron de la primera fase. Los cuales fueron: desarrollo ágil de software, ingeniería de software, sistemas de información, desarrollo lean de software, desarrollo de software, métodos agiles, industria del software, y metodología tradicional. Se recuperaron los documentos comprendidos en el periodo de 2010 a octubre 2020. En los casos donde la información era similar entre los artículos se seleccionó el documento que contuviera más citas según sus bases de datos o el índice de la revista. De ese segundo análisis se obtuvo un total de 28 artículos.

METODOLOGIAS TRADICIONALES

En sus inicios, en la década de los 70´s, el desarrollo de software era un proceso totalmente artesanal, a través de metodologías tradicionales. Éstas eran efectivas al cumplir con la función de gestión y desarrollo de sistemas de software. Sin embargo, esta industria comenzó a sufrir cambios por la necesidad constante de modificar los requerimientos del proyecto y la falta de involucramiento por parte de los usuarios del sistema. Estos inconvenientes conllevaron a la necesidad de mejorar el proceso y dirigir los proyectos hacia la meta planteada. Las primeras aportaciones fueron extraídas de conceptos y metodologías ya existentes en otras áreas como la ingeniería, construcción y automotriz. Por ejemplo, el proceso del desarrollo del software estaba dividido en etapas de manera secuencial, lo que mejoró la necesidad latente de adaptación e innovación en el campo del software (Figueroa, Solís y Cabrera, 2016).

Al respecto, una Metodología de Desarrollo de Sistemas, (SDM) es una colección documentada de políticas, procesos y procedimientos, para mejorar la técnica de desarrollo de software en términos de una mayor productividad del personal de Tecnología de la Información (TI) y una mayor calidad de la información (Chan y Thong, 2009).

Las metodologías tradicionales se caracterizan por tener una estructura de desarrollo claramente establecida, lineal y poco flexible ante cambios de un entorno volátil. Presentan un alto costo al ser implementadas. Tienen roles, actividades y artefactos claramente definidos. Requieren de una documentación exhaustiva, clara y detallada de todo el proyecto. Y centran su atención en cumplir un plan establecido desde el inicio del desarrollo, por todas las partes involucradas. Por ejemplo: RUP, MSF, cascada, espiral, ICONIX, entre otros.

Sin embargo, a lo largo de los años, los requerimientos de los clientes cambiaron; ahora se tienen restricciones de tiempo, debe existir flexibilidad a lo largo del desarrollo del proyecto y en lo posible una disminución de costos. Por lo que, los equipos de desarrollo tienden a prescindir de la implementación de estas metodologías (Parada, 2016).

Investigadores y profesionales comenzaron a buscar métodos alternativos para la implementación de proyectos, ante el reconocimiento de que los modelos tradicionales de planificación y ejecución de desarrollo de software no eran óptimos o no estaban ajustados a los desafíos específicos que enfrentan los proyectos. De hecho, es debido a estos desafíos que las técnicas de gestión de proyectos

"ligeras" ganaron y conservan su popularidad desde que se desarrollaron por primera vez (Dybå y Dingsøyr, 2008).

Cabe mencionar que las metodologías tradicionales hasta el día de hoy son utilizadas en el desarrollo de software por su implementación de buenas prácticas, control y seguimiento de cada aspecto que las conforma. Pero ante las ventajas que brinda una metodología ágil en cuestión de tiempo y dinero se han visto desfavorecidas. Tanta es la presión ejercida sobre los desarrolladores que incluso los llevó a adoptar y en algunos casos a crear nuevas metodologías de desarrollo.

Se tienen dos estudios que dan evidencia de algunas dificultades que enfrentan los desarrolladores de software al utilizar metodologías tradicionales. Fitzgerald (1996) informó que el 50 % de las actividades de diseño ocurrieron en fases distintas a las establecidas. Por lo tanto, el problema crítico que enfrentaban los gerentes eran el desajuste entre el deseo de congelación temprana de especificaciones y planes fijos con la necesidad concomitante de mantener suficiente flexibilidad para modificar y alterar los planes del proyecto para abordar las necesidades comerciales críticas. Asimismo, en un estudio realizado por Collyer et al. (2010) encontraron que las metodologías tradicionales tenían dificultades en entornos dinámicos debido a tres tipos de cambios que ocurrían con frecuencia en los proyectos: 1) metas; 2) materiales, recursos, herramientas y técnicas; y 3) relaciones con otros proyectos, servicios o productos relacionados.

METODOLOGÍAS ÁGILES

El desarrollo de metodologías ágiles evolucionó a partir de las experiencias personales y colectiva de los consultores y líderes de opinión de la comunidad del software. Si bien la mayoría de las prácticas ágiles individuales tienen un atractivo intuitivo, ya que se basan en principios de gestión con aceptación general, ciertamente carecían de fundamentos teóricos o de apoyo empírico para sus beneficios declarados, cuando se concibieron inicialmente. Por lo que, existía una necesidad urgente de perspectivas teóricas que arrojaran luz sobre cómo estas prácticas dispares y sus interacciones podían producir resultados valiosos. Es así como el comprender teóricamente la distinción entre métodos ágiles y sus contrapartes tradicionales fue una preocupación que pidió atención a la investigación (Dingsøyr et al., 2012).

Tras una reunión de creadores e impulsores del desarrollo de software, celebrada en Utah, Estados Unidos en 2001, se crea The Agile Alliance. Organización dedicada a promover los "Métodos Ágiles". Los métodos ágiles eran una alternativa a los métodos ya existentes. Los cuales, que como ya se mencionó previamente, eran pesados y rígidos por su carácter normativo, dirigidos por documentación exhaustiva, fuerte dependencia de planificaciones detalladas y que resultaban alejados a las necesidades de los clientes. También, se crea el "Manifiesto Ágil" constituido por cuatro postulados que buscan un cambio de mentalidad y una nueva cultura organizativa. Asimismo, se crean los 12 principios que permiten desarrollar software de manera rápida y responder efectivamente a los cambios que surgen a lo largo del proyecto (Chan y Thong; 2009; Letelier et al.,2003).

Las metodologías ágiles son una nueva serie de metodologías que pretenden superar las limitaciones de los SDM tradicionales basados en planes. Su uso en el desarrollo de software se ha extendido en los últimos años, toda vez que permite contrarrestar los peligros de los métodos tradicionales de planificación inicial que a menudo conducen a patologías de desarrollo posteriores (Serrador y Pinto, 2015).

Existe una gran variedad de metodologías ágiles que permiten desarrollar productos de software de calidad, y la planeación y control del proyecto. Por ejemplo: eXtremeProgramming (XP), Crystal Methods (CM), Adaptive Software Development (ASD), Agile Modeling (AM), Agile RUP, DynamicSolutionsDeliveryModel (DSDM), Evolutionary Project Management (EVO), Feature-DrivenDevelopment (FDD), entre otras.

Estos métodos incorporan los principios del Manifiesto Ágil. Primero, hubo un movimiento distintivo hacia el desarrollo colaborativo. Las personas tienen privilegios sobre los procesos que antes los limitaban. Segundo, se abogó por una mentalidad "esbelta" para minimizar el trabajo innecesario en la documentación derrochadora. Es decir, documentar solo lo que era absolutamente necesario y nada más. Tercero, los clientes/partes interesadas ya no se encontraban al margen del desarrollo de software, sino que dan forma y guían la evolución del producto final. Cuarto, se aceptó que la incertidumbre era parte del desarrollo de software, y que la tendencia a controlar las variaciones a través de medios estadísticos era inútil (Dingsøyr et al., 2012).

Snowden en 1999 introduce el Marco Cynefin (“Hábitat”), que sirve para identificar el tipo de metodología ágil a implementar en el desarrollo del software. Este marco se originó en la práctica de la gestión del conocimiento, con apoyo de diversas áreas: sistemas, redes, estrategia, complejidad, aprendizaje y dirección. Es definido como un “sense-making framework”, o marco conceptual utilizado para ayudar en la toma de decisiones. Asimismo, es un medio para distinguir entre comunidades formales e informales, y para la interacción de ambos con procesos estructurados y condiciones inciertas. A demás ayuda a identificar las problemáticas y reconocer las técnicas adecuadas para tratarlas y finalmente, para determinar la viabilidad de implementación de dichas metodologías (Kurtz y Snowden, 2003).

El marco Cynefin compara las características de cinco contextos con diferentes grados de complejidad: simple, complicado, complejo, caótico y desordenado. Asimismo, se plantea la importancia que tiene para los lideres identificar y definir el contexto de las situaciones en las que se trabajará. Lo cual, ayuda a diseñar las acciones a seguir a fin de optimizar los recursos materiales, de infraestructura, económicos y humanos. Es así como, de acuerdo con Snowden y Boone (2007) los contextos pueden ser descritos de la siguiente manera:

O. A) Los contextos simples se caracterizan por la estabilidad y relaciones de causa y efecto claras para cualquier persona. A menudo la respuesta correcta es autoevidente. Es aquí donde “se sabe lo que se sabe”. Los líderes deben percibir los hechos de una situación, categorizar y responder.
P. B) Los contextos complicados pueden contener muchas respuestas correctas, y aunque hay una clara relación entre causa y efecto, no todas las personas pueden verlas. Aquí “se sabe lo que no se sabe”. Los líderes deben percibir, analizar y responder.
Q. C) En los contextos complejos, es imposible hallar respuestas correctas. Más bien, los patrones instructivos emergen si el líder realiza experimentos donde se puede fallar sin riesgo. Es el contexto donde “no se sabe lo que no se sabe”, y en este escenario es donde operan muchas de las empresas. Los líderes deben sondear, percibir y responder.
R. D) En un contexto caótico, buscar respuestas correctas es inútil. Las relaciones entre causa y efecto no se pueden determinar debido a que cambian constantemente y no existen patrones claros. Es el contexto de “lo que no se puede saber”. El líder debe actuar para establecer el orden, percibir dónde hay estabilidad y trabajar para cambiar la situación de caos a complejidad.
S. E) El quinto contexto, desorden, ocurre cuando no está claro cuál de los otros contextos es el predominante. La salida es dividir la situación en sus partes constitutivas y asignar cada una a los otros reinos. Entonces, los líderes pueden tomar decisiones en forma adecuada a su contexto.

FACTORES DE ÉXITO DE LAS METODOLOGIAS ÁGILES

El desarrollo de software es una actividad de suma importancia en cualquier aspecto de la vida de las personas. Sin embargo, Chow y Cao (2008) sostienen que, a pesar de los esfuerzos por emplear metodologías de software, tanto ágiles como tradicionales, su desarrollo no ha tenido un éxito constante, por lo que a menudo resulta en proyectos de software con retraso, fallidos, en abandono o en rechazo. Incluso algunos proyectos ya implementados pueden necesitar un costoso mantenimiento continuo, versiones correctivas o paquetes de servicios.

Las metodologías ágiles a menudo son aceptadas tanto por los gerentes como por los programadores, ya que proporcionan una liberación de gastos que son impuestos por los enfoques tradicionales de desarrollo de software. Sin embargo, en la práctica, pocas organizaciones son capaces psicológica o técnicamente, de adoptar estas metodologías de forma inmediata o implementarlas con éxito en un corto tiempo. Además, puede ser inapropiado que sean completamente ágiles en todos los aspectos del desarrollo, quizás deban retener elementos bien conocidos y confiables de un enfoque más tradicional dentro de un proyecto ágil general y todavía existen preocupaciones sobre la capacidad de utilizarlos en proyectos grandes y en entornos complejos (Qumer y Henderson-Sellers, 2008a).

Al respecto, se menciona que el fracaso o éxito de los proyectos que se gestionan a través de metodologías agiles radica en la complejidad del software y especialmente del tamaño del equipo de trabajo involucrado en su desarrollo. Un proyecto grande tiene al menos 2 a 9 equipos y cada equipo tiene de 5 a 9 miembros (Dingsøyr et al., 2014) y un proyecto de gran tamaño tiene en uno de los equipos 20 o más miembros (Hannay y Benestad, 2010). Actualmente, no existe consenso sobre cómo se debe

desarrollar un proyecto de gran tamaño, toda vez que puede ser a través de métodos tradicionales, ágiles o híbridos (VersionOne, 2018; Theocharis et al., 2015).

Los factores que determinan el éxito de las metodologías ágiles versan en cuatro dimensiones: calidad, alcance, tiempo y costo. Los factores de incidencia son (a) una estrategia de entrega correcta, (b) una práctica adecuada de técnicas de ingeniería de software ágil y (c) un alto nivel de rendimiento (equipo de calibre). Otros tres factores que podrían ser críticos para ciertas dimensiones de éxito son: (a) un buen proceso de gestión de proyectos ágil, (b) un entorno de equipo amigable con ágil y (c) una fuerte participación del cliente. No se tiene evidencia de que algunos prerrequisitos asumidos para el éxito de los proyectos ágiles sean, un fuerte apoyo ejecutivo, un fuerte compromiso del patrocinador, la disponibilidad inmediata de instalaciones ágiles físicas o tipos de proyectos apropiados para desarrollo ágil, etc. (Chow, y Cao, 2008).

Por su parte Misra, Kumar y Kumar, (2009) establecen que los factores de éxito son la satisfacción, colaboración y compromiso del cliente, tiempo de decisión, cultura corporativa, control, características personales, cultura social, formación y aprendizaje. No se tiene evidencia de que factores como: distribución y tamaño del equipo, planificación, competencia técnica, comunicación y la negación tuvieran una relación significativa con el éxito.

Un estudio realizado en empresas de telecomunicaciones que utilizan la metodología Scrum muestra la relación que existe entre los factores de personalidad y el clima laboral del equipo ágil. Existe una correlación positiva significativa entre la apertura a la experiencia y el apoyo a la innovación. La amabilidad se correlaciona positivamente con el clima general del equipo. Los rasgos de personalidad representan menos del 15 % de la varianza en el clima del equipo (Vishnubhotla, Mendes, y Lundberg, 2020).

Un caso de éxito en la implementación de estas metodologías agiles fue el realizado por Azanha et al. (2017) en una empresa farmacéutica brasileña. El estudio comparó Scrum con el modelo tradicional en cascada. El análisis reflejó beneficios en la utilización del marco ágil: mayor motivación y satisfacción del personal, mejor control de los requisitos, mayor calidad del sistema que se entregó. Lo cual, generó valor agregado para la organización. El proyecto permitió el uso de funciones desde el primer mes de implementación de la aplicación. Asimismo, una reducción del 75 % en el tiempo de desarrollo, en comparación con los métodos tradicionales, en un tiempo de cuatro meses, el 30 por ciento de lo que sería el total si se adoptara la metodología tradicional.

COMPARACIÓN ENTRE METODOLOGÍAS ÁGILES

En función de lo establecido se vuelve inevitable que entre las metodologías ágiles también surjan comparaciones que denoten las ventajas y desventajas que cada una posee y que permite a los desarrolladores tomar una decisión más consiente sobre la más apropiada o la que más se acomode a las necesidades individuales. A continuación, los resultados de la comparación de SCRUM y XP, Crystal y Delfdroid.

Parada (2016) exploró el uso de las metodologías ágiles en el desarrollo de software móvil sobre la base de 1,234 egresados que laboran en diferentes empresas. Los resultados arrojaron que la metodología SCRUM es la más usada con el 43,42 %, XP con 10,50 %, y Crystal con el 9.2 %. También, resultó interesante la decisión de hacer uso de una metodología hibrida entre Scrum/XP con el 19,70 %. Este estudio también proporciona información concerniente a las características que los profesionales consideran importantes para tener en cuenta en el momento de elegir una metodología ágil: que tengan los roles claramente definidos 97.4 %, tamaño del equipo desarrollador 97.4 %, la comunicación constante con el cliente 97.4 %, y permite realizar entregas del producto 90.8 %.

Ávila & Meneses (2013) compararon la metodología SCRUM vs XP vs Delfdroid, esta última pertenece a una categoría hibrida entre las dos primeras. El estudio se basó en tres parámetros: vista del sistema como algo cambiante, tener en cuenta la colaboración entre los miembros del equipo y características más específicas de la propia metodología como son simplicidad, excelencia técnica, resultados, adaptabilidad, etc. Como resultado Delfdroid se benefició de la fusión de las mejores características de cada una de las otras metodologías.

METODOLOGÍAS TRADICIONALES VS ÁGILES

Diversos autores desarrollan estudios que comparan las metodologías ágiles vs tradicionales. Dyba y Dingsøyr (2008) afirman que las primeras ponen a los individuos e interacciones sobre los procesos y

herramientas de las metodologías tradicionales. Al software que funciona sobre la documentación exhaustiva. La colaboración con el cliente sobre la negociación de contratos y la respuesta ante el cambio sobre el seguimiento de un plan.

Fitzgerald (1996) encontró que el 60 % de las organizaciones encuestadas no usaban metodologías, mientras que solo el 6 % seguía una metodología rigurosamente y el 79 % de las que no usaban ninguna metodología no tenía la intención de adoptar una. Algunas posibles razones de la baja aceptación son: (1) los desarrolladores simplemente ignoran los SDM recién introducidos, (2) los SDM tratan el proceso de desarrollo de sistemas de forma racional y ordenado cuando no es así y (3) se supone que los SDM deben ser de aplicación universal y ajustarse a diferentes situaciones de desarrollo.

Letelier et al. (2003) afirman que las metodologías tradicionales se encuentran basadas en normas estandarizadas, presentan resistencia al cambio, tienen un proceso muy controlado, la interacción del cliente es mediante reuniones, hay más artefactos y más roles que provocan grupos de trabajo grandes y énfasis en la arquitectura del software. Mientras que la metodología ágil se encuentra basada en heurísticas provenientes de prácticas de producción de código, diseñada para cambios repentinos, un proceso menos controlado, el cliente forma parte del equipo de desarrollo, hay pocos artefactos, pocos roles, en consecuencia, grupos pequeños de desarrollo y presentan menos énfasis en la arquitectura del software.

Apiumhub (2018) proporciona las siguientes estadísticas: la metodología ágil mejora en 54 % la colaboración entre equipos que no acostumbran a trabajar juntos. Aumento del 52 % en la calidad del software y 49 % en la satisfacción del cliente con el producto. Disminución de un 43 % en el tiempo de comercialización y un 42 % en el costo del desarrollo. El estudio concluye que los desarrolladores de software actuales muestran una tendencia superior a la implementación de las metodologías ágiles sobre las metodologías tradicionales por las ventajas que esta primera representa tanto para el equipo de desarrollo como para el cliente.

CONCLUSIÓN

Las metodologías tradicionales se han visto rebasadas por las actuales demandas de la industria como son: adaptación a las expectativas de clientes y usuarios, tiempos y costos de desarrollo y requerimientos susceptibles a cambios.

La aplicación efectiva de las metodologías ágiles representa beneficios tanto para los miembros del equipo de desarrollo como para los clientes. Tales como reducción de costos, capacidad de predicción conforme se entregan las iteraciones, trabajo colaborativo y mayor satisfacción y motivación cliente – equipo. Todo esto desde una filosofía que jerarquiza individuos e interacciones sobre procesos y herramientas.

Las diferencias entre los SDM tradicionales y ágiles conducen a discrepancias en una serie de prácticas y requisitos como la planificación y el control, la asignación de roles entre desarrolladores, el rol del cliente y la tecnología utilizada. Los estudios realizados sobre estas metodologías de desarrollo aún no favorecen a ninguna. A pesar de las diversas ventajas potenciales que brindan en general los SDM, los profesionales presentan cierta resistencia a la adopción de dichas metodologías.

Actualmente existen numerosas metodologías agiles, y la elección de estas por los directivos y desarrolladores dependerá de las necesidades, familiaridad o comodidad que el usuario o usuarios presenten al implementarla. Asimismo, se están desarrollando e implementando metodologías hibridas entre tradicionales-ágiles y ágiles-ágiles, que pretenden combinar los mejores factores de las ya existentes para facilitar aún más su usabilidad e implementación en la industria del desarrollo de software.

Referencias

1. Apiumhub. (**2018**). *Transformación ágil; pasos, estadísticas y un ejemplo práctico.* https://apiumhub.com/es/tech-blog-barcelona/transformacion-agil-pasos-estadisticas/

2. Ávila, E., & Meneses, A. (**2013**). *Delfdroid y su comparación evaluativa con XP y Scrum mediante el método 4-DATRevista Cubana de Ciencias Informáticas,* 7(1), 2227–1899. http://rcci.uci.cu

3. Azanha, A., Argoud, A. R. T. T., de Camargo Junior, J. B., y Antoniolli, P. D. (**2017**). *Agile project management with Scrum: A case study of a Brazilian pharmaceutical company IT projectInternational Journal of Managing Projects in Business,* 10(1), 121–142 https://doi.org/10.1108/IJMPB-06-2016-0054

4. Canós, J., Letelier, P., y Penadés, M. (**2006**). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Técnica Administrativa,* 5(26), 1.

5. Chan, F. K. Y., y Thong, J. Y. L. (**2009**). *Acceptance of agile methodologies : A critical review and conceptual framework. Decision Support Systems*46(4), 803–814. https://doi.org/10.1016/j.dss.2008.11.009

6. Chow, T., y Cao, D(**2008**). *A survey study of critical success factors in agile software projects. The Journal of Systems and Software 81 (2008)* 961–971, 81, 961–971 https://doi.org/10.1016/j.jss.2007.08.020

7. Collyer, S., Warren, C., Hemsley, B., & Stevens, C. (**2010**). *Aim, fire, aim - Project planning styles in dynamic environments. Project Management Journal*41(4), 108–121. https://doi.org/10.1002/pmj.20199

8. Dingsøyr, T., Fægri, T., y Itkonen, J. (**2014**). What is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8892(2014), 273–276. https://doi.org/10.1007/978-3-319-13835-0

9. Dingsøyr, T., Nerur, S., Balijepally, V., y Moe, N. B. (**2012**). *A decade of agile methodologies : Towards explaining agile software development. The Journal of Systems and Software,* 85, 1213–1221. https://doi.org/10.1016/j.jss.2012.02.033

10. Dyba, T., y Dingsøyr, T. (**2008**). *Empirical studies of agile software development : A systematic review. Information and Software Technology 50 (2008)* 833–859, 50, 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

11. Figueroa, R., Solís, C., y Cabrera, A. (**2016**). *Metodologías Tradicionales Vs Metodologías Ágiles. Universidad Técnica Particular de Loja*Escuela de Ciencias en Computación, 1–9.

12. Fitzgerald, B. (**1996**). *Formalized systems development methodologies: A critical perspective. Information Systems Journal*6(1), 3–23. https://doi.org/10.1111/j.1365-2575.1996.tb00002.x

13. Hannay, J. E., y Benestad, H. C. (**2010**). *(2010). Perceived productivity threats in large agile development projectsESEM 2010 - Proceedings of the 2010 ACM-IEEE International Symposium on Empirical* Engineering and Measurement, 1325 https://doi.org/10.1145/1852786.1852806

14. Kurtz, C., y Snowden, D. (**2003**). *The new dynamics of strategy : Sense-making in a complex and complicated world. Diario de sistemas de IBM*42(3), 462–483 https://doi.org/10.1147 / sj.423.0462

15. Letelier, P., Penadés, M., Canós, J., y Sánchez, E (**2003**). *Metodologías Ágiles en el Desarrollo de Software. Universidad Politecnica de Valencia.*

16. Misra, S., Kumar, V., & Kumar, U. (**2009**). *Identifying some important success factors in adopting agile software development practices. The Journal of Systems & Software,* 82(11), 1869–1890. https://doi.org/10.1016/j.jss.2009.05.052

17. Parada, C. (**2016**). *Caracterización de las metodologias agiles para el desarrollo de aplicaciones moviles. Universidad Francisco de Paula Santander*, 1–6.

18. Qumer, A. y Henderson-Sellers, B. (**2008**). *A framework to support the evaluation, adoption and improvement of agile methods in practice. The Journal of Systems & Software 81 (2008* 1899–1919, 81, 1899–1919
https://doi.org/10.1016/j.jss.2007.12.806

19. Serrador, P., y Pinto, J. K. (**2015**). *Does Agile work ? — A quantitative analysis of agile project success. International Journal of Project Management 33 (2015)*1040–1051, 33(5), 1040–1051.
https://doi.org/10.1016/j.ijproman.2015.01.006

20. Snowden, D., y Boone, M. (**2007**). *Un marco para la toma de decisiones del líder. Harvard Business Review,* 85(11), 122–132.

21. Theocharis, G., Kuhrmann, M., Münch, J., & Diebold, P. (**2015**). *Is water-scrum-fall reality? On the use of agile and traditional development practices. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* 9459, 149–166
https://doi.org/10.1007/978-3-319-26844-6_11

22. VersionOne. (**2018**). *The 12th annual state of agile survey. 12 Th Annual State of Agile Development Survey.*
https://www.versionone.com/about/press-releases/12th-annual-state-of-agile-survey-open/

23. Vishnubhotla, S. D., Mendes, E., y Lundberg, L. (**2020**). *Investigating the relationship between personalities and agile team climate of software professionals in a telecom company. Information and Software Technology*126(March).
https://doi.org/10.1016/j.infsof.2020.106335