ESTABLISHMENT OF AGILE TEAMS FOR SOFTWARE DEVELOPMENT: REVIEW OF LITERATURE[i]

CONFORMACIÓN DE EQUIPOS ÁGILES PARA EL DESARROLLO DE SOFTWARE: REVISIÓN DE LITERATURA

FORMAÇÃO EQUIPAMENTO AGILE DESENVOLVIMENTO DE SOFTWARE

FABIOLA SÁENZ BLANCO[ii]
FREDERICK GUTIÉRREZ SIERRA[iii]
JULIÁN CAMILO RAMOS RIVERA[iv]

CITACIÓN

Sáenz-Blanco, Fabiola; Gutiérrez-Sierra, Frederick & Ramos-Rivera, Julián C. & (2017). Establishment of agile teams for software development: review of literature. *Dimensión Empresarial*, 16(2), 39-54. DOI: http://dx.doi.org/10.15665/dem.v16i2.1901

JEL: M12, M15

ABSTRACT

This review article aims to analyze one of the most important elements in the software industry: the conformation of self-managed work teams and the implementation of a planning, control and development methodology of software project. For that, a literature review was made, researching journal articles, graduation projects and books, among others, to know the software industry generalities worldwide, as well as the situation in Colombia, the conformation of self-managed work teams and the agile methodologies features for planning, executing and controlling development projects. The self-managed teams' conformation eases communication and knowledge sharing, essential elements inside the software development industry since they allow the generation of an appropriate answer in front of the changes caused by the environment. Supporting on the Scrum development methodology, value for the stakeholders is obtained and a workflow is set, guided to improve project productivity and efficiency.

Keywords: Software Industry, Agile Methodology, Scrum, Self-Managed Teams.

RESUMEN

El presente artículo de revisión tiene como finalidad analizar uno de los elementos más importantes en la industria de software: La conformación de equipos autogestionados y la implementación de una metodología para la planeación, control y desarrollo de los proyectos de software. Se realizó una revisión bibliográfica en diferentes bases de datos académicas, con el fin de conocer las generalidades y estado de la industria de software, la conformación de equipos autogestionados y las características de las metodologías ágiles de desarrollo. La conformación de equipos autogestionados facilita la comunicación y transmisión de conocimiento, elementos fundamentales dentro de la industria de desarrollo, permitiendo generar una respuesta adecuada frente a los cambios ocasionados por el entorno. Apoyándose en la metodología de desarrollo *Scrum*, se obtiene valor para los Stakeholders y se establece un flujo de trabajo encaminado a mejorar la productividad y eficiencia del proyecto.

Palabras clave**:** Industria de Software, Metodología Ágil, Scrum, Equipos Autogestionados.

RESUMO

Este artigo é uma revisão e análise de um dos elementos mais importantes da indústria de software: A criação de equipes auto-geridas e implementação de uma metodologia de planejamento, controle e desenvolvimento de projetos de software. Uma revisão da literatura em diferentes bancos de dados acadêmicos foi realizada a fim de conhecer generalidades e estado da indústria de software, a criação de equipamentos de auto-gestão e as características de metodologias ágeis de desenvolvimento. A formação de equipes auto-geridas facilita a comunicação e transmissão de conhecimento, elementos fundamentais dentro da indústria de desenvolvimento, permitindo gerar uma resposta adequada às mudanças causadas pelo ambiente. Com base na metodologia de desenvolvimento Scrum, o valor das partes interessadas é obtido e um fluxo de trabalho projetado para melhorar a produtividade e eficiência do projeto é estabelecida.

Palavras-chave**:** Indústria de software, metodologia Agile, Scrum, equipes auto-geridas.

INTRODUCCIÓN

La industria del software representa actualmente uno de los mercados más importantes y competitivos en el mundo, principalmente en Estados Unidos y Alemania, con una fuerte participación emergente de países como India, Irlanda, Israel, Brasil, China, entre otros (Palomino, 2011). En Colombia esta participación no es la excepción, pero aún no se tiene el nivel competitivo suficiente para ser un referente mundial, principalmente debido a dificultades como lo son: el 95% de los empresarios son ingenieros de sistemas sin formación administrativa, financiera o comercial adecuada, informalidad en los procesos, la desconfianza de clientes potenciales ante empresas emergentes, la escasa inversión nacional y/o extranjera, el bajo nivel de asociatividad del gremio, entre otros (Díaz & Ospina, 2014b).

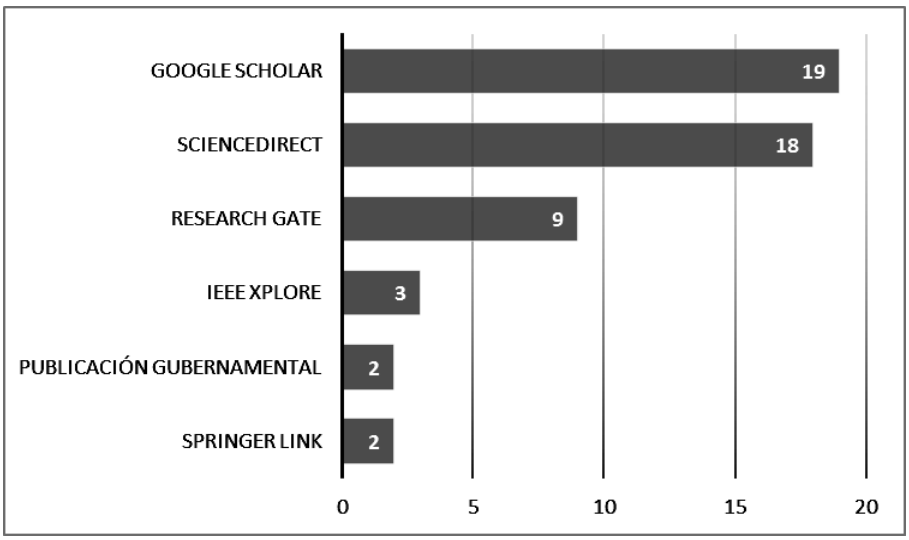
Adicionalmente, se presentan otros problemas inherentes a las empresas emergentes, tales como: intentos de realizar un buen trabajo de desarrollo sin contar con procesos bien definidos, la baja o nula implementación de una metodología que gestione no sólo el proceso desarrollo de software sino a toda la organización, y la utilidad y eficiencia asociada a prácticas dentro del proceso creativo que normalmente varían entre las diferentes organizaciones o proyectos, por lo que su estandarización no es posible (Pedroza, 2013). Como consecuencia de lo anteriormente descrito, los proyectos de software pueden presentar retrasos, ser fallidos, rechazados o abandonados. En caso de ser implementados, pueden requerir mantenimientos costosos durante su ejecución o incluso lanzamientos correctivos que reemplacen el producto inicial (Chow & Cao, 2008).

Este artículo tiene como fin presentar una revisión de los estudios realizados sobre la industria de software tanto a nivel global como específico para Colombia, y la manera en cómo los proyectos obtienen mejores resultados al preferir la conformación de Equipos Autogestionados y los beneficios de la implementación de Metodologías de Gestión de Proyectos Ágiles frente a las Metodologías Tradicionales. Estos temas son abordados de la siguiente manera: En la sección 2 se describe la metodología usada para la revisión de literatura y el planteamiento de las preguntas que originaron la necesidad del presente texto; en la sección 3 se trata la industria de software a nivel general y las característica más relevantes del sector Colombiano; en la sección 4 abordamos cómo la conformación de Equipos Autogestionados favorece la transmisión de información y por ende, el incremento de productividad; en las secciones 5, 6 y 7 se exponen las diferentes metodologías de gestión de proyectos: tradicionales, ágiles, y se enfatiza en la metodología ágil Scrum, respectivamente. Finalmente, las secciones 8 y 9 muestran la discusión de los autores y las conclusiones.

METODOLOGÍA

Para el desarrollo del artículo, se realizaron consultas en las siguientes bases de datos y herramientas de investigación: *Science Direct*, *IEEE Xplore*, *Springer Link*, *Research Gate*, *Google Scholar*, además de fuentes gubernamentales en temas específicos al estado del país en el sector de Tecnologías de la información.

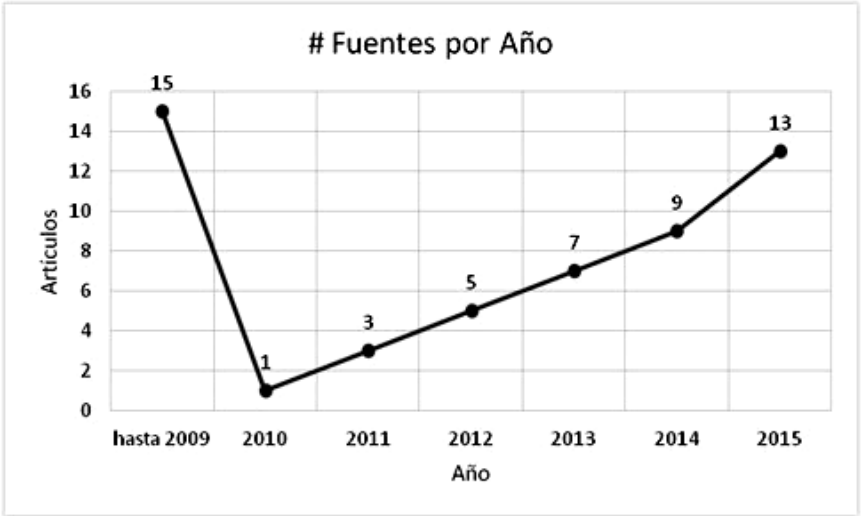
Figura 1. Distribución de fuentes para consulta Bibliográfica



Fuente: Autores

La revisión documental permite identificar que la mayor cantidad de artículos consultados se encuentran en la franja de tiempo entre 2012 y 2015, con un total de 34 fuentes. Durante este periodo, se identifica un alza en el número de investigaciones relacionadas con la industria de desarrollo de software, los equipos de desarrollo, la gestión de proyectos de software y las metodologías ágiles, permitiendo inferir que existe una oportunidad de mejora en dichos aspectos en las empresas del sector y que está siendo considerada como elemento diferenciador que otorgaría cierta ventaja competitiva. Los artículos comprendidos en el periodo de tiempo entre 1986 y 2009 dan una visual inicial del estado del arte y permiten sentar una base en la investigación documental presentada en este escrito.

Figura 2. Distribución de fuentes por año de publicación



Fuente: Autores

Para la sistematización del proceso de recopilación de información fueron usadas las siguientes palabras clave de acuerdo con los diferentes tópicos de interés. Estos se expresan en la Tabla 1.

Tabla 1. Palabras claves de búsqueda

Categoría	Palabra de búsqueda
Desarrollo de Software	Software, software development, software engineering, software development process
Metodologías Ágiles de desarrollo	Agile, agile methodologies, scrum, extreme programming, project management, requirement management, user stories
Equipos de desarrollo	Software development teams, self-organized, knowledge sharing, cooperation, knowledge management, social media, autonomy

Fuente: Autores

Las palabras clave de búsqueda y los temas a los cuales se encuentran asociados buscan darle respuesta a las preguntas que han motivado la creación del presente artículo; ¿Cuál es la caracterización de la industria de desarrollo en Colombia?, ¿Cómo debe ser el manejo de equipos de desarrollo?, ¿Cuál es el mejor modo para gestionar proyectos de software?, ¿Cuál sería el marco de trabajo adaptado a las necesidades específicas de la industria colombiana?

LA INDUSTRIA DEL DESARROLLO DE SOFTWARE

El desarrollo de software es un proceso de producción no tradicional, que no involucra materias primas, ni maquinaria especializada. Es, en cambio, una actividad netamente intelectual y creativa, en donde el principal recurso es el capital humano con un alto nivel de especialización en diferentes tecnologías y herramientas que permita una adecuada ejecución del proyecto. Sin embargo, la presencia o no de dicho personal no constituye una barrera de entrada al mercado para nuevas compañías.

Existen diversas metodologías tradicionales para la creación de un producto-servicio, pero se podría considerar la expuesta por Kruchten (Kruchten, 1996) como base para entender las generalidades del proceso de desarrollo. El Proceso Unificado Racional (RUP: Rational Unified Process) define cuatro fases: inicio, elaboración, desarrollo y transición (Figura 3). Otros autores indican que las fases tradicionales del ciclo de vida de un programa son: Requerimiento de Ingeniería, Especificación, Arquitectura, Diseño, Implementación, Testeo, Mantenimiento y Evolución (Figura 4). (Woodcock, Larsen, Bicarregui, & Fitzgerald, 2009) (Perkusich, Soares, Almeida, & Perkusich, 2015).

Adicionalmente, el desarrollo de cualquier producto de software o relacionado con TI, no acaba con la entrega de un producto final, sino que generalmente existe una fase de soporte dada la curva de obsolescencia (2 a 3 años) a la que deben enfrentarse la mayoría de los proyectos lo cual hace necesario un continuo lanzamiento de actualizaciones y mejoras (Sommerville, 2010). sumando a esto los aspectos más sensibles para los clientes que contratan servicios de desarrollo de software y TI son el soporte post venta, el precio y la innovación que tenga el producto a entregar: elementos que impactan actualmente la industria y están condicionando la competitividad de las compañías, tanto en el interior como en el exterior (Díaz & Ospina, 2014a).

Figura 3. Fases del Proceso Unificado Racional RUP

Fuente: Autores con base en Kruchten (1996)

En términos generales, las principales barreras de entrada en el mercado globalizado para las industrias de software son: las condiciones internas del país, que favorecen o no la diversificación industrial y las exportaciones, el tamaño del mercado local, el tamaño de la empresa, las pocas posibilidades de financiarse ante la negativa de los bancos, la inexistencia de adecuados procesos de aseguramiento de la calidad, y la alta variabilidad de los costos, dependiendo del tipo de producto que se quiera lanzar (Correa, 1996).

En el panorama colombiano, cabe resaltar el crecimiento continuo de las exportaciones de servicios de software y TI que han venido aumentando desde 2008 a una tasa promedio de 32% anual. Como resultado, para el 2014 la participación de este sector en el PIB Nacional era de alrededor del 3,2%, donde la participación por empresas era de: 43% empresas medianas, 37% empresas pequeñas, 19% empresas grandes y solo el 1% microempresas; demostrando una consolidación del sector en el país. No obstante, de los ingresos operacionales generados ese año, el 72% se concentró en las grandes empresas (Superintendencia de Sociedades, 2015).

Esta distribución tan marcada de los ingresos se debe en gran medida al tiempo requerido para la ejecución de proyectos, principalmente de gran tamaño, que en Colombia es en promedio de 12 a 18 meses para su desarrollo, dependiendo de los requerimientos (Díaz & Ospina, 2014b). Solo las grandes y medianas empresas cuentan con los recursos para mantener la ejecución de un proyecto de tal magnitud, mientras que empresas pequeñas y microempresas deben ajustarse a proyectos de corto tiempo, que representen rápidamente ganancias, con el fin de dar continuidad a su operación.

En Colombia, los desarrollos de software iniciaron de manera "artesanal", para los cuales no se seguía ningún estándar de calidad ni metodologías de desarrollo. Con la llegada de nuevas tecnologías, plataformas de desarrollo y programas de formación superior mejor estructurados, se inició un proceso de mejoramiento de procesos entre los que se incluyó la planeación y seguimiento de los proyectos de software (Palomino, 2011), dado que existe un consenso general al considerar que la principal capacidad de Colombia para desarrollar y fortalecer el desarrollo de software es el recurso humano (ESICenter SinerTIC Andino, 2012).

Figura 4. Fases del ciclo de vida de los programas de software

Fuente: Autores con base en Woodcock (2009)

Con el fin de apalancar el desarrollo nacional es necesario realizar un fortalecimiento técnico interno, aumentar la comercialización y despegue de nuevos proyectos, e incrementar el intercambio de conocimiento adecuado entre los diferentes actores del sector (ESICenter SinerTIC Andino, 2012). Además, se deben generar condiciones que propicien un efectivo proceso de innovación, que resulte en el desarrollo de nuevos productos o servicios, crecimiento organizacional y ventaja competitiva, factores clave para una adecuada incursión en el mercado de desarrollo de software.

EQUIPOS AUTOGESTIONADOS: MODELO EFECTIVO DE INTERCAMBIO DE CONOCIMIENTO

De acuerdo con el consenso general de diferentes expertos y organizaciones, Colombia tiene como ventaja competitiva el recurso humano disponible (altamente capacitado, o que puede ser capacitado) (ESICenter SinerTIC Andino, 2012). Sin embargo, dada la naturaleza del proceso de desarrollo de software que involucra elementos creativos y de trabajo en equipo altamente especializados, la manera en cómo se conformen dichos equipos, su capacidad para comunicarse efectivamente y las herramientas que tengan a su alcance impactarán indudablemente en el éxito del proceso.

Uno de los factores críticos a la hora de iniciar un proyecto es el nivel de interacción entre los distintos actores involucrados en el desarrollo de este (desarrolladores, Stakeholders), además de la metodología que se implemente para generar avances e identificar problemas durante el proceso (Takeuchi & Nonaka, 1986). Un intercambio efectivo de conocimiento es necesario para permitir que los miembros del equipo discutan aspectos críticos del proyecto, razón por la cual se debe garantizar, en la medida de lo posible, la efectividad en la socialización (Ghobadi, 2015).

El desarrollo de software es un proceso colaborativo en el que su éxito depende de la adquisición de conocimiento, el intercambio de información y la minimización de la ruptura en el proceso de comunicación (Kusumasari, Supriana, Surendro, & Sastramihardja, 2011). Los equipos multifuncionales se componen de representantes individuales extraídos de varias unidades funcionales, como departamentos u organizaciones, que poseen conocimiento especializado y habilidades relevantes para la realización y terminación de un proyecto (Janz & Prasarnphanich, 2009). Bajo esta perspectiva, los equipos de desarrollo de software multifuncionales son grupos de trabajo temporales que cargan con la responsabilidad de terminar un proyecto de desarrollo dentro de un marco limitado de tiempo, y que son conformados por miembros representativos traídos desde varias unidades funcionales (Ghobadi & D'Ambra, 2013).

En muchos casos estos equipos se encuentran dispersos globalmente (ya sea por motivos de reducción de costos, o especialización de algún tema particular), lo cual genera retos y riesgos específicos que deben ser considerados, como son: la dispersión geográfica, rupturas en el proceso de control y coordinación, pérdida de comunicación, pérdida del espíritu de trabajo en equipo y diferencias culturales (Portillo, Vizcaíno, Piattini, & Beecham, 2012). Si bien Colombia es un buen lugar para la conformación de equipos multidisciplinares, un escenario común es el ingreso de compañías multinacionales que cuentan con equipos alrededor del mundo. Para mitigar los impactos ocasionados por situaciones similares se ponen a disposición de los equipos diferentes herramientas como SoSo (Social Software); que son canales no estructurados de comunicación y conocimiento y permiten complementar los métodos formales de comunicación dentro de la compañía, dentro de los cuales vale la pena destacar:

- IM: Instant Messaging, permite una comunicación punto a punto que resulta muy efectiva en relación a su costo. En diferentes estudios ha demostrado incrementar la productividad de los empleados, permitiendo realizar preguntas específicas sobre aspectos laborales. Genera resultados más rápidos que el correo electrónico en equipos distribuidos en diferentes ubicaciones, da lugar a respuestas rápidas y feedbacks inmediatos (Giuffrida & Dittrich, 2013).
- Blog y Microblog: Usados como herramienta para compartir conocimiento organizacional y específico, permite la distribución del mismo a lo largo de la compañía. Suele estar compuesto de pequeñas frases, hipervínculos, videos e imágenes. Al contrario de una Wiki permite expresar opiniones más personales. Su principal reto es el cambio cultural que impone generar contenido. No muchas personas lo adoptan fácilmente (Giuffrida & Dittrich, 2013).
- Wiki: Funciona como un gran repositorio de conocimiento, en donde se busca que la compañía tenga fácil acceso a información de diferente índole. Aunque es una herramienta muy poderosa, requiere de mucho más tiempo de administración que un blog, dado que se debe estructurar el contenido, realizar un glosario, mantenerlo actualizado, entre otras actividades que pueden tomar un tiempo considerable (Giuffrida & Dittrich, 2013).
- IDE: Entornos integrados de desarrollo, facilitan la elaboración de aplicativos web y móviles, puesto que ofrecen herramientas de edición, compilación, depuración, análisis y ejecución en una misma interfaz, mejorando los tiempos de ejecución de las etapas de diseño, desarrollo y pruebas de la metodología que se esté usando (Gasca, Camargo, & Medina, 2014).
- Otras herramientas comúnmente usadas son redes sociales y social bookmarking (Giuffrida & Dittrich, 2013).

Facilitando la comunicación de los miembros de los equipos (sin importar su ubicación), se comienza a

reducir los riesgos e incertidumbres propias de un trabajo colaborativo entre personas. Ahora bien, estas personas presentan una gran cantidad de interacciones con sus pares que pueden desencadenar un conjunto variado de emociones y/o sentimientos. Algunas tan negativas para el desarrollo del proyecto como la frustración y el disgusto, que afectan la calidad del producto; u otras con resultados tan inesperados como la ira, que según estudios generan efectos positivos en la productividad de un desarrollador.

La continua retrospectiva del equipo permite que existan alternativas ante diferentes dificultades que se puedan presentar. Muchas metodologías de desarrollo tienen herramientas que miden la satisfacción del cliente final con el producto, sin embargo, pocas o ninguna tienen en consideración el estado de los desarrolladores. Se debe tener en cuenta la correlación existente entre los sentimientos del equipo de desarrollo y la productividad asociada al mismo (Jurado & Rodríguez, 2015). Es, entonces, donde cobra especial importancia el uso de metodologías no tradicionales que tomen en consideración aspectos acuciantes para la ejecución del proyecto.

UNA TRADICIÓN COMO METODOLOGÍAS DE GESTIÓN

Las metodologías tradicionales (cuyo uso impera en el desarrollo de software en el país) no resultan óptimas para adaptarse a los cambios que enfrentan los proyectos (Patel et al., 2012). Usualmente, tienen dificultades con temas críticos como lo son: El cumplimiento de metas; el manejo de material, recursos, herramientas y técnicas, y las interacciones con otros proyectos, servicios o productos que guardan relación (Serrador & Pinto, 2015). Este proceso de desarrollo de software hace énfasis en el control de los procesos por medio de la definición precisa de roles, actividades, recursos y artefactos, incluyendo los modelos de diseño, bien documentados y detallados (Pedroza, 2013). Esta gran carga de labores entorpece el proceso de planeación y ejecución durante un proyecto de desarrollo de software (Merchán, Urrea, & Rebollar, 2008).

Esta gestión clásica considera un proyecto exitoso cuando este se ha completado a tiempo, en el presupuesto y con las funcionalidades definidas al principio (Torrecilla, Sedeño, Escalona, & Mejías, 2015). No da lugar a una evaluación constante del producto que se realiza y solo tiene en cuenta la ejecución de pruebas como una actividad que se lleva a cabo una vez terminada la fase de codificación y que tiene como propósito identificar fallos (Yagüe & Garbajosa, 2009).

Normalmente este esquema tradicional para la gerencia de proyectos de software tiene su principal participación en proyectos de gran tamaño, donde se exige normalmente un alto nivel de burocracia o procedimientos asociados (Letelier & Penadés, 2006), dando lugar a la implementación del Modelo de cascada, que es una serie de pasos seguidos en forma secuencial, de acuerdo a lo que se planeó al inicio del proyecto. El proceso de desarrollo se divide en subtareas, donde no se pasa de una tarea a otra hasta que el resultado de la primera es evaluado y aprobado. Este modelo supone un problema para la flexibilidad del proceso de desarrollo (Domann, Hartmann, Burkhardt, Barge, & Albayrak, 2014). Las empresas necesitan de metodologías que soporten todas las actividades que efectivamente se realicen, y que pueden ser el resultado inesperado de cualquier serie de interacciones tanto internas como externas a la ejecución del plan de trabajo (Patel et al., 2012).

Este conjunto de inconvenientes, asociados al uso de metodologías tradicionales, y las grandes dificultades que presenta el sector colombiano de desarrollo de software y TI, evidencia la necesidad de considerar la implementación de una metodología más acorde a las necesidades de la industria, que responda rápidamente a los cambios que se puedan presentar, y permita mantener a los proyectos dentro de un margen de

ganancias posterior a su ejecución (Rujana, Franco, Tortosa, & Tomaselli, 2016).

AGILIDAD EN LA GESTIÓN DE PROYECTOS

Las metodologías ágiles están soportadas en un conjunto de principios relacionados, resumidos en el Manifiesto Ágil, que se enfocan en dar prioridad a los elementos que agregan más valor durante el desarrollo del proyecto (Blakeman, 2008). Estos principios son: el individuo y la interacción del equipo por encima del proceso y las herramientas; el software funcional por encima de la documentación; la colaboración con y del cliente por encima de la negociación del contrato; la respuesta a los cambios por encima del seguimiento a un plan (Abrahamsson, et al, 2002, 107; Letelier & Penadés, 2006; Pedroza, 2013).

Para que una metodología de desarrollo de software convencional sea considerada flexible, debe permitir que los interesados en el proyecto (Stakeholders) seleccionen los elementos que más valor aporten al mismo, evitando la realización de actividades innecesarias y acentuando las que resulten más significativas (Diez, Britos, Rossi, & García-Martínez, 2003). Los aspectos en donde más contrasta un método ágil con respecto a un método tradicional son: Énfasis en el diseño continuo, alcance flexible manejo de la incertidumbre y alta interacción con los clientes (Serrador & Pinto, 2015).

El Desarrollo Ágil de Software ha sido adoptado extensamente por la industria gracias a la evidencia empírica obtenida al aplicar estas metodologías en diferentes organizaciones (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003). La creciente importancia, complejidad y demanda, requiere que los equipos de desarrollo se encuentren trabajando de la mano con los potenciales usuarios, los clientes y los expertos en el tema (Yip & Juhola, 2015). Esto es logrado a través de continuo monitoreo y estimación, además de una frecuente intervención, que permita obtener una mayor respuesta a los rápidos cambios del sector, tanto de tecnología como de requerimientos (Torrecilla et al., 2015).

Si bien la implementación de metodologías ágiles corresponde a la acogida que han tenido sus más populares exponentes, que son XP (Extreme Programming) y Scrum, cabe resaltar que los equipos más exitosos han implementado prácticas ágiles de distintas metodologías para generar un proceso de desarrollo propio que se adapte a sus necesidades y requerimientos particulares (Navarro, Fernández, & Morales, 2013) (Ahmad, Baharom, & Husni, 2012). Cuando compañías que cuentan con una larga trayectoria de metodologías tradicionales deciden realizar su paso a una metodología ágil, deben tener especial cuidado, poniendo mayor énfasis en compartir la cultura ágil entre los diferentes miembros y atacando cada problema conforme se vaya presentando (Papadopoulos, 2015).

En general, las metodologías ágiles buscan beneficiar al equipo y aumentar la productividad tanto como sea posible. Así se implementan procesos y principios como: programación en parejas, refactorización, integración continua, 40 horas por semana de trabajo máximo, estándares de programación, entre otros muchos elementos que definen un marco de trabajo en beneficio del producto y del empleado (Letelier & Penadés, 2006).

Los proyectos que se trabajan bajo una metodología ágil inician con un proceso de descubrimiento en donde se define el alcance y la naturaleza del desarrollo. Sus fases incluyen la declaración de la visión, la jerarquización de características, la estimación aproximada de la inversión y la ficha de datos del proyecto (Blakeman, 2008). En la medida que un sistema de software se hace cada vez más grande, la gestión de requerimientos se hace cada vez más desafiante. El control sobre los requerimientos facilita la anticipación y la respuesta ante solicitudes de cambio, siendo esta una de las actividades más críticas para el éxito del desarrollo de software y el ciclo de vida del

proyecto (Baruah, 2015).

El desarrollo de software implica el reto de planear actividades mientras el sistema de software evoluciona y sus características aumentan (Domann et al., 2014), sobre la marcha se aprende del producto, del cliente y del equipo. Algunas metodologías ágiles usan historias de usuario para definir los requerimientos que se tienen, ya sea que existan desde el principio, o que se hayan ido agregando conforme se avanza en la ejecución del proyecto (Izaurralde, 2013).

Una historia de usuario puede definirse como una pequeña parte de funcionalidad que le brinda algún tipo de valor al cliente, al usuario o al sistema. Las historias de usuario representan una necesidad que puede no estar documentada, sino que posiblemente ha sido descubierta durante el proceso colaborativo (Torrecilla et al., 2015). Las historias de usuario suelen tener un formato similar al siguiente: "Como [usuario, cliente, administrador, etc.] quiero poder [Funcionalidad] para [Resultado]" (Estayno & Meles, 2014; Suaza, 2013).

Se busca con las historias de usuario validar las necesidades y dirigir la implementación. No obstante, una de las principales debilidades de la captura de información mediante historias de usuario es que normalmente son realizadas por el cliente bajo un lenguaje "comprensible" para todos. Este tipo de documentación no da cabida a un proceso posterior a la entrega del proyecto (Escobar, Velandia, Ordoñez, & Cobos, 2015). En cambio, si el lenguaje usado para la creación de las historias de usuario es definido en conjunto, y resulta lo suficientemente claro para todos los involucrados en el proyecto, y junto con esto se tienen las pruebas que se realicen para la conformidad del requerimiento, se llega a tener una guía para el posterior desarrollo del software, ya sea en fases de soporte o actualización (Yagüe & Garbajosa, 2009).

SCRUM, COMO METODOLOGÍA INTEGRAL

Scrum es un sistema iterativo que estructura el desarrollo en ciclos de trabajo llamados Sprints. Cada iteración tiene una duración no mayor a un mes y se basa en la motivación del trabajo en equipo y una ágil reacción al cambio, con miras al resultado final. Es especialmente adecuado para proyectos cuyos requerimientos cambian rápidamente y existe incertidumbre al respecto (Barrios et al., 2012). Esta metodología, adoptada en muchas empresas y corporaciones dedicadas al desarrollo de software o actividades relacionadas con TI, es actualmente el marco de trabajo más popular en la gestión de proyectos ágiles. Se pretende realizar una aproximación a sus características y las bondades que representa para las organizaciones, tanto a nivel colaborativo entre los miembros del equipo, como en la dirección y correcta ejecución de planes de trabajo.

Scrum promueve la formación de los equipos ágiles que se caracterizan por ser autogestionados, multifuncionales y trabajar en iteraciones. La autogestión les permite elegir la mejor manera de realizar el trabajo, dado que los integrantes del equipo tienen todos los conocimientos necesarios para el cumplimiento del proyecto (Navarro et al., 2013), y más específicamente, de las actividades consignadas en el Backlog del producto, un artefacto usado en la metodología para registrar los requerimientos, funcionalidad y necesidades del proyecto (Palacio, 2014).

Dado el alto nivel de responsabilidad y empoderamiento de los equipos, es posible aumentar continuamente el valor de los productos de acuerdo con las necesidades y requerimientos de los Stakeholders, puesto que cada nueva integración conlleva un conjunto de entregables encaminados a generar valor (Navarro et al., 2013). El contenido de cada iteración, es decir, las actividades a realizar, es consensuado por el equipo, previo al inicio de la

iteración o Sprint, en una reunión que generalmente se conoce como Sprint Planning (Barrios et al., 2012).

Dentro de esta metodología, son muy importantes las diferentes reuniones del equipo, que cuentan con el objetivo de generar coordinación e integración entre los diferentes actores del proceso de desarrollo. Las reuniones son:

- Sprint Planning: El equipo, el Scrum Master y el Product Owner se reúnen para definir cuál es la meta de la iteración y qué actividades conllevarán a alcanzar esta meta. El equipo determina cuánto pueden lograr dentro de la iteración. Para este momento el Backlog de tareas se encuentra priorizado, lo cual permite que, al momento de seleccionar las tareas a realizar, se están tomando las que más valor generan al cliente.
- Daily Scrum: Es una reunión de no más de 15 minutos, en donde el equipo se reúne y cada persona responde 3 preguntas. ¿Qué se ha logrado desde el último Daily Scrum?, ¿Qué se logrará hasta el próximo Daily Scrum?, ¿Qué impedimentos tiene para lograr el cumplimiento de sus tareas? Estas preguntas buscan mantener la incertidumbre del proyecto al mínimo y propiciar colaboración entre los miembros del equipo.
- Sprint Review: El equipo presenta los logros alcanzados durante el Sprint. El Product Owner revisa las implementaciones y soluciones aportadas y estas son aceptadas, o se agregan nuevos ítems o historias al Backlog.
- Sprint Retrospective: Es una reunión que solo involucra al equipo y al Scrum Máster. Se busca en esta reunión analizar 3 aspectos relacionados con la iteración que finaliza: qué estuvo bien durante el Sprint, qué no, y qué mejoras podrían hacerse para el siguiente sprint. El Sprint Retrospective es un parte integral del proceso de mejora continua y cambio ágil (Abrahamsson et al., 2002,).

El continuo control, revisión y mejoramiento, resultado de la ejecución adecuada de estas reuniones, permiten mantener la flexibilidad y rápida respuesta ante los cambios que se presenten, omitiendo la necesidad de generar documentación detallada, de carácter burocrático (Fertalj & Katic, 2008). Toda la documentación que se requiere en el proyecto está atada a unas buenas prácticas de codificación, unas historias de usuario claras y entendibles por todos los involucrados en el negocio, y un set de pruebas satisfactorio y detallado (González, Dominguez, Gutiérrez, & Escalona, 2014).

Además de los beneficios mencionados anteriormente, en un estudio realizado por Mann y Maurer (Mann & Maurer, 2005) se encontró que, para los clientes, las reuniones diarias permiten mantener un estado actualizado que disminuye la incertidumbre y la confusión sobre los elementos a desarrollar y aquellos que no, dado que finalmente no aportan valor agregado al producto (Serrador & Pinto, 2015).

No obstante, las reuniones por sí solas no determinan la efectividad de la metodología. Para que se efectúe un adecuado proceso de ejecución, se deben realizar estimaciones correctas donde la planeación de cada iteración debe resultar acertada. En términos generales, en los métodos ágiles se utilizan prácticas empíricas basadas en juicios de expertos para estimar el costo de desarrollo para un nuevo producto (Mitre, Ortega, & Lemus, 2014).

Para realizar estimaciones más precisas se debe considerar la velocidad del equipo, el principal indicador del Scrum, que determina la cantidad de trabajo que el equipo puede realizar satisfactoriamente durante la duración de un Sprint (la velocidad puede estar dada en cualquier unidad de medida que el equipo considere conveniente, sean puntos de historia, puntos de funcionalidad, entre otros). Tener claridad sobre este indicador permite que los equipos realicen su trabajo de una manera adecuada (sin la presión de fechas imposibles de cumplir), teniendo en cuenta factores críticos propios del desarrollo de software. como lo son la adquisición de deuda técnica y la entrega de valor

(Cogollo, 2013).

Dentro de la metodología Scrum se cuenta con un conjunto de roles que buscan el éxito del proyecto. El primero de los roles es el Product Owner, siendo este responsable del éxito del producto desde el punto de vista de los Stakeholders. Debe contar con las capacidades para determinar la visión y expectativas, realizar la recolección de requerimientos y funcionalidades, y priorizar continuamente al Backlog de requerimientos (Cogollo, 2013). Para ello debe evaluar la importancia de acuerdo con la proximidad (grado de colaboración y/o cooperación, compromiso y frecuencia en el proceso desarrollo de software) y la participación (comunicación entre el Stakeholder y el equipo de desarrollo, incluyendo instrucciones o reglas dadas al equipo) (Yip & Juhola, 2015).

Además del Product Owner, existe otro rol dentro de la metodología llamado Scrum Máster, cuyo objetivo es velar por el cumplimiento de los principios ágiles y Scrum. Para ello debe educar al Product Owner y al Scrum Team, y tomar en consideración los diferentes aspectos que son críticos para la organización, facilitando la comunicación y el trabajo del equipo, e impidiendo que se presenten interferencias externas durante la ejecución de una iteración (Sprint) (Barrios et al., 2012).

DISCUSIÓN

Con el fin de garantizar que los proyectos de software sean exitosos y, a su vez, incrementar la productividad de las compañías de desarrollo de software o actividades relacionadas con TI, se requiere la implementación adecuada de una metodología ágil, siendo Scrum la opción más viable como deja ver Barrios (Barrios et al., 2012), dados sus sencillos pasos de implementación y los altos niveles de adaptabilidad que tiene.

Esta adaptabilidad genera, en consecuencia, condiciones que favorecen la formación de equipos autogestionados, en los que una comunicación continua y efectiva permite el desarrollo y la evolución de los proyectos. Es importante recalcar que, aunque la autogestión es inherente a la correcta implementación de una metodología ágil, conlleva un esfuerzo considerable lograr y mantener las condiciones propicias para generar el empoderamiento por parte de los miembros de un equipo. Por ello se recalca la necesidad de propiciar una comunicación asertiva que permita compartir fácilmente el conocimiento, mejore la calidad del trabajo en equipo y su vez la calidad técnica del producto; y que, en consecuencia, impacte de manera favorable la coordinación entre los miembros, su contribución al proyecto desde las diferentes áreas de experticia, la cohesión y el soporte a cada uno de los integrantes (Lindsjørn, Sjøberg, Dingsøyr, Bergersen, & Dybå, 2016).

Sin embargo, no se puede simplemente aseverar que la metodología recomendada resulta efectiva para cualquier caso o proyecto en específico. Es importante que se consideren la mayor cantidad de variables posibles y se busque un punto en donde, sin importar cuál sea la metodología implementada para la gestión de proyectos, se logre satisfacer los requerimientos y necesidades del cliente. Špundak (Špundak, 2014) sugiere que se maneje una combinación de metodologías tradicionales y agiles, en donde prime, la coherencia con otros procesos que desarrolle la compañía y que para la mayoría de los casos resulte en metodologías ajustadas a cada empresa; entendiendo sus limitaciones y también como se aprovechan diferentes herramientas para obtener los mejores resultados.

La naturaleza de los proyectos de software, resulta ser un determinante importante a considerar dentro de la evaluación previa a la selección o construcción de metodología a implementar y aunque existe un fuerte debate entre las bondades y desventajas de las diferentes aproximaciones para la gestión de proyectos, se debe buscar propiciar una rápida respuesta a los

cambios que se presenten, además de respuestas proactivas y reactivas con el fin de obtener conocimiento que pueda traducirse a futuro en valor para el cliente (Dingsøyra, Nerur, Balijepally, & Moe, 2012).

Jurado y Rodríguez recalcan la necesidad de continuas evaluaciones al equipo con respecto a su satisfacción con los procesos que se realizan internamente (Jurado & Rodríguez, 2015), así como Giuffrida y Dittrich hacen lo mismo con la promoción de herramientas de SoSo (Giuffrida & Dittrich, 2013), y Kusumasari hace otro tanto enfatizando la importancia de la comunicación asertiva y oportuna entre los miembros del equipo (Kusumasari et al., 2011). La unión de estos tres aspectos sugiere una mejora importante en el proceso de desarrollo de software, aspecto que debería considerarse fundamental en empresas emergentes.

Por tanto, la adopción de la metodología adecuada se encuentra sujeta a las diferentes condiciones de cada compañía, organización o grupo funcional. Por lo cual, se deja abierta la puerta a un proceso de investigación que valide la efectividad de las empresas colombianas que han adoptado una metodología ágil, con los lineamientos necesarios para la conformación de equipos autogestionados que favorezcan la transmisión de conocimiento, frente a compañías que continúan realizando procesos bajo metodologías tradicionales, o no implementen nada más que soluciones producto de la experiencia propia.

CONCLUSIONES

El dinamismo del macrosistema en el que se encuentra inmerso el desarrollo de software ha permitido la creación de distintas metodologías que han sido útiles según las condiciones del momento en que fueron establecidas. En consecuencia, conforme cierta metodología no satisface las condiciones necesarias para la supervivencia de una compañía en el mercado, se rompe el paradigma (como en el caso del Manifiesto Ágil) y el proceso se adapta para satisfacer los nuevos requerimientos de los clientes. Es así como se ha pasado de una metodología tradicional de tipo lineal (RUP, Cascada) a unas más sistémicas como las metodologías ágiles (XP, Scrum, etc.).

Las metodologías ágiles resultan benéficas para las organizaciones en la medida en que permiten que el desarrollo de software sea un proceso de mejoramiento continuo, cooperativo entre el usuario y el desarrollador, sencillo de aprender y con respuesta rápida ante los cambios (Serrador & Pinto, 2015). Pero para llegar a este nivel dinámico, es importante que la conformación e integración del equipo de desarrollo favorezca un buen ambiente de trabajo, una comunicación asertiva y un intercambio adecuado de conocimiento.

Dado que el método Scrum no requiere ninguna práctica específica de ingeniería, puede adaptarse a la mayor cantidad de condiciones que presente una organización (algo bastante benéfico para un entorno cambiante y complejo como el de la industria colombiana) (Abrahamsson et al., 2002, 107). La implementación de esta metodología dentro de una organización requiere una participación por parte del Scrum Máster como facilitador de procesos, que comparta la cultura ágil entre los diferentes miembros del equipo. El principal cambio que se debe lograr para implementar correctamente el marco de trabajo Scrum es la correcta definición de las funcionalidades o tareas que se realizarán en cada iteración y la adopción de una cultura enfocada a generar valor para los Stakeholders.

Un proceso de implementación exitoso resultará en un equipo autogestionado que no dependa del Scrum Máster para la correcta aplicación de los principios, que tenga un alto nivel de comunicación y que haya refinado sus estimaciones hasta el punto en que puedan cumplirse a cabalidad, iteración tras iteración (Barrios et al., 2012).

REFERENCIAS

Abrahamsson, Pekka; Salo, Outi; Ronkainen, Jussi & Warsta, Juhani (2002). Agile Software Development Methods: Review and Analysis. VTT Publications 478. En: https://www.vtt.fi/inf/pdf/publications/2002/P478.pdf. [17/11/2016] (fecha en que se bajó y leyó)

Abrahamsson, P., Warsta, J., Siponen, M., & Ronkainen, J. (2003). New directions on agile methods: A comparative analysis. 25th Int. Conference On Software Engineering, Proc., 244–254.

Ahmad, F., Baharom, F., & Husni, M. (2012). Agile Development Methods for Developing Web Application in Small Software Firms. Knowledge Management Inrenational Conference (KMICe), (4–6 July), 281–285.

Barrios, W. G., Godoy Guglielmone, M. V., Fernández, M. G., Mariño, S. I., Ferreira, F. M., & Zarrabeitia, C. T. (2012). SCRUM: application experience in a software development PyME in the NEA. Journal of Computer Science & Technology, 12(3), 110–115.

Baruah, N. (2015). Requirement management in agile software environment. Procedia Computer Science, 62, 81–83. DOI: https://doi.org/10.1016/j.procs.2015.08.414

Blakeman, G. (2008). Implementing Agile Software Development for Small – Medium Business. The University of Kansas.

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. Journal of Systems and Software, 81(6), 961–971.

Cogollo, J. (2013). Análisis, propuesta y representación de indicadores en proyectos ágiles con SCRUM. Cuaderno Activa, (5), 11–21.

Correa, C. M. (1996). Strategies for software exports from developing countries. World Development, 24(1), 171–182.

Díaz, M., & Ospina, M. (2014a). Identificación de variables que afectarían la valoración de productos de las micro, pequeñas y medianas empresas dedicadas al desarrollo de software por encargo en Colombia. Informador Técnico (Colombia), 78(1), 64–81.

Díaz, M., & Ospina, M. (2014b). Prospectiva 2019 - 2023 para Mipymes dedicadas al desarrollo de software por encargo en Colombia. El Hombre Y La Máquina, (44), 75–91.

Diez, E., Britos, P., Rossi, B., & García-Martínez, R. (2003). Generación Asistida del Mapa de Actividades de Proyectos de Desarrollo de Software. Reportes Técnicos En Ingeniería Del Software, 5(1), 13–18.

Dingsøyra, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, 85(6), 1213–1221.

Domann, J., Hartmann, S., Burkhardt, M., Barge, A., & Albayrak, S. (2014). An agile method for multiagent software engineering. Procedia Computer Science, 32(DAAF), 928–934.

Escobar, A., Velandia, D., Ordoñez, H., & Cobos, C. (2015). A review of the impact on XP methodology of business model inclusion in requirements elicitation. Sistemas & Telemática, 13(33), 45–61.

ESICenter SinerTIC Andino. (2012). *Proyecto generación de estrategias para el desarrollo tecnológico y de mercado del sector software y servicios de ti mediante la aplicación de vigilancia tecnológica y prospectiva*. En: http://cenisoft.org/wp-content/uploads/sites/2/2016/08/Estudio_Vigilancia_Tecnologica_Y_Prospectiva.pdf [25/11/2016]

Estayno, M., & Meles, J. (2014). El Rol del Product Owner en la definición y validación de las user stories. Ciencia Y Tecnología, 14, 145–162.

Fertalj, K., & Katic, M. (2008). An overview of modern software development methodologies. 19th Central European Conference on Information and Intelligent Systems, 633–639.

Gasca, M., Camargo, L., & Medina, B. (2014). Metodología para el desarrollo de aplicaciones móviles. Tecnura, 18(40), 20–35.

Ghobadi, S. (2015). What drives knowledge sharing in software development teams: A literature review and classification framework. Information & Management, 52(1), 82–97.

Ghobadi, S., & D'Ambra, J. (2013). Modeling High-Quality Knowledge Sharing in cross-functional software development teams. Information Processing and Management, 49(1), 138–157.

Giuffrida, R., & Dittrich, Y. (2013). Empirical studies on the use of social software in global software development – A systematic mapping study. Information and Software Technology, 55(7), 1143–1164.

González, J., Dominguez, F., Gutiérrez, J., & Escalona, M. (2014). Pruebas de aceptación orientadas al usuario: Contexto ágil para un proyecto de gestión documental. Ibersid, 8, 73–80.

Izaurralde, M. (2013). Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario.

Universidad Tecnológica Nacional.
Janz, B. D., & Prasarnphanich, P. (2009). Freedom to Cooperate: Gaining Clarity Into Knowledge Integration in Information Systems Development Teams. IEEE Transactions on Engineering Management, 56(4), 621–635.
Jurado, F., & Rodriguez, P. (2015). Sentiment Analysis in monitoring software development processes: An exploratory case study on GitHub's project issues. Journal of Systems and Software, 104, 82–89.
Kruchten, P. (1996). A Rational Development Process. Crosstalk, 9(July), 11–16.
Kusumasari, T., Supriana, I., Surendro, K., & Sastramihardja, H. (2011). Collaboration Model of Software Development. 2011 International Conference on Electrical Engineering and Informatics, 79, 1–6.
Letelier, P., & Penadés, M. (2006). Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Técnica Administrativa, 5(26), 1–14.
Lindsjørn, Y., Sjøberg, D. I. K., Dingsøyr, T., Bergersen, G. R., & Dybå, T. (2016). Teamwork quality and project success in software development: A survey of agile development teams. Journal of Systems and Software, 122, 274–286.
Mann, C., & Maurer, F. (2005). A case study on the impact of Scrum on overtime and customer satisfaction. Proceedings of the Agile Development Conference (ADC'05).
Merchán, L., Urrea, A., & Rebollar, R. (2008). Definición de una metodología ágil de ingeniería de requerimientos para empresas emergentes de desarrollo de software del sur-occidente colombiano. Revista Científica Guillermo de Ockham, 6(1), 37–50.
Mitre, H., Ortega, E., & Lemus, C. (2014). Estimación y control de costos en métodos ágiles para desarrollo de software: un caso de estudio. Ingeniería, Investigación Y Tecnología, 15(3), 403–418.
Navarro, A., Fernández, J., & Morales, J. (2013). Revisión de metodologías ágiles para el desarrollo de software. Prospectiva, 11(2), 30–39.
Palacio, J. (2014). Gestión de proyectos Scrum Manager. (R. Claudia, Ed.) (1st ed.). Safe Creative.
Palomino, K. (2011). Estudio del comportamiento de la industria del software en Colombia ante escenarios de capacidades de innovación y ventajas competitivas por medio de dinámica de sistemas. Universidad Nacional de Colombia.
Papadopoulos, G. (2015). Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects. Procedia - Social and Behavioral Sciences, 175, 455–463.
Patel, A., Seyfi, A., Taghavi, M., Wills, C., Na, L., Latih, R., & Misra, S. (2012). A comparative study of agile, component-based, aspect-oriented and mashup software development methods. Technical Gazette, 19(1), 175–189.
Pedroza, P. (2013). Elección de una Metodología de Desarrollo a partir de las Ventajas de una Metodología Ágil y un Modelo Robusto como CMMI-DEV 1.3. Ingeniare, 8(14), 113–122.
Perkusich, M., Soares, G., Almeida, H., & Perkusich, A. (2015). A procedure to detect problems of processes in software development projects using Bayesian networks. Expert Systems with Applications, 42(1), 437–450.
Portillo, J., Vizcaíno, A., Piattini, M., & Beecham, S. (2012). Tools used in Global Software Engineering: A systematic mapping review. Information and Software Technology, 54(7), 663–685.
Rujana, M., Franco, N. R., Tortosa, N., & Tomaselli, G. (2016). Análisis sobre adopción de metodologías ágiles en los equipos de desarrollo en pymes del NEA. XVIII Workshop de Investigadores En Ciencias de La Computación, 646–650.
Serrador, P., & Pinto, J. (2015). Does Agile work ? - A quantitative analysis of agile project success. International Journal of Project Management, 33, 1040–1051.
Sommerville, I. (2010). Software Engineering. Software Engineering.
Špundak, M. (2014). Mixed Agile/Traditional Project Management Methodology – Reality or Illusion? Procedia - Social and Behavioral Sciences, 119, 939–948.
Suaza, K. V. (2013). Definición de equivalencias entre historias de usuario y especificaciones en UN - LENCEP para el desarrollo ágil de software. Universidad Nacional de Colombia Facultad.
Superintendencia de Sociedades. (2015). Desempeño Del Sector Software 2012 - 2014.
Takeuchi, H., & Nonaka, I. (1986). The new new product development game. Journal of Product Innovation Management, 3(3), 205–206.
Torrecilla, C., Sedeño, J., Escalona, M., & Mejías, M. (2015). Estimating , planning and managing Agile Web development projects under a value-based perspective. Information and Software Technology, 61, 124–144.
Woodcock, J., Larsen, P., Bicarregui, J., & Fitzgerald, J. (2009). Formal methods: Practice and Experience. ACM Comput. Surv., 41(4), 1–36.

Yagüe, A., & Garbajosa, J. (2009). Comparativa práctica de las pruebas en entornos tradicionales y ágiles. Revista Española de Innovación, Calidad E Ingeniería Del Software (REICIS), 5(4), 19–32.

Yip, M. H., & Juhola, T. (2015). Stakeholder involvement in software system development - Insights into the influence of product-service ratio. Technology in Society, 43, 105–114.

NOTAS

[i] Artículo de revisión de literatura enfocado en los temas de: Industria de software en Colombia y el mundo, conformación de equipos autogestionados, y metodologías de desarrollo de software con énfasis en metodologías ágiles, gestado en el grupo de investigación ARCOSES (Arquitectura del Conocimiento y Sistemas Expertos) de la Universidad Distrital Francisco José de Caldas, Bogotá, Colombia. www.udistrital.edu.co . Fecha de recepción: [14/12/2017], fecha de aceptación [04/03/2018]

[ii] Ingeniera Industrial, PhD, Docente Titular de Carrera, Investigadora grupo ARCOSES, Universidad Distrital Francisco José de Caldas, www.udistrital.edu.co, Bogotá, Colombia, correo: fsaenz@udistrital.edu.co.

[iii] Estudiante de Ingeniería Industrial, Scrum Máster certificado por el Scrum Institute, Universidad Distrital Francisco José de Caldas, www.udistrital.edu.co, Bogotá, Colombia, correo: fgutierrezs@correo.udistrital.edu.co.

[iv] Estudiante de Ingeniería Industrial, Universidad Distrital Francisco José de Caldas, www.udistrital.edu.co, Bogotá, Colombia, correo: jcramosr@correo.udistrital,edu.co