¿Por qué implementar Scrum?

Why to implement Scrum?

Pour quelles raisons mettre en place Scrum?

Porque implementar Scrum?

César Rodríguez*
Rubén Dorado**

Fecha de recepción: 15 de abril de 2015
Fecha de aprobación: 10 de junio de 2015
Pp.125-144.

* Especialista en Gerencia de Proyectos, Universidad del Rosario. Ingeniero de Sistemas, Universidad Nacional de Colombia. Certificado como Scrum Master, Agile Institute. Certificado PMP, Project Management Institute. Certificado ITIL Foundation, Axelos.

** Magíster en Ciencias de la Computación, Universidad de Tokyo. Ingeniero de Sistemas, Universidad Nacional de Colombia.

Resumen

En este artículo se discuten algunos argumentos del por qué en algunos casos, al momento de seleccionar una metodología de desarrollo de *Software* Scrum, puede ser la mejor alternativa, exponiendo los principales diferenciadores de esta con respecto a otras del mercado. Así mismo, aborda desde un punto de vista general, la evolución de los esquemas en los cuales se basan las diferentes metodologías de *Software* resaltando los principales inconvenientes que han tenido y el porqué de la necesidad de buscar nuevos modelos. Finalmente, se destacan los beneficios de SCRUM como una metodología diferenciadora entre otras de desarrollo ágil, particularmente *eXtreme Programming*.

Palabras clave

Metodologías de *Software,* casos de desarrollo de *Software*, administración de proyectos de *Software*, metodologías de desarrollo ágiles.

Abstract

This article discusses some arguments why in some cases, selecting a SCRUM Software development methodology may be the best alternative, presenting the key differentiators between this methodology and others in the market. Likewise, it addresses from a general point of view, the development of schemes in which different Software methodologies are based, disclosing the main problems they have had and the need to seek new models. Finally, the benefits of Scrum as a differentiating methodology are highlighted among other methodologies of agile development, particularly eXtreme Programming.

Key words

Software methodologies, Software Development Cases, Software Project Management, Agile Development Methodologies.

Résumé

Dans cet article nous présentons un certains nombre d'arguments démontrant que lors de la sélection d'une méthodologie de conception de logiciels, la méthodologie Scrum peut être la meilleure alternative et nous analyserons les principales différences de ce logiciel par rapport à ces concurents présents sur le marché. De même, nous étudierons l'évolution des schémas sur lesquels se basent les différentes methodologies de logiciels et montrerons les principaux inconvénients existants ainsi que les raisons de la nécessité d'une recherche de nouveaux modèles. Nous montrerons finalement les bénéfices de SCRUM comme méthodologie d'eXtreme Programming pour une conception efficace et différente.

Mots clefs

Méthodologie de logiciel, cas de conception de software, gestion de projets de logiciel, méthodologies de conception efficientes

Resumo

Neste artigo se discutem alguns argumentos do porquê em alguns casos, ao momento de seleccionar uma metodologia de desenvolvimento do Software Scrum, pode ser a melhor alternativa, expondo os principais diferenciadores desta em relação a outras do mercado. Assim mesmo, aborda a partir de um ponto de vista geral, a evolução dos esquemas nos quais se baseiam as diferentes metodologias de Software ressaltando os principais inconvenientes que tem tido e o porquê da necessidad de procurar novos modelos. Finalmente, se destacam os benefícios de SCRUM como uma metodologia diferenciadora entre outras de desenvolvimento ágil, particularmente Extreme Programming.

Palavras-chave

Tempo de secado, verniz, catalizador, diluente, Anova, fatores, níveis, variável de resposta, produtividade, eficiência.

1. Introducción

El campo del desarrollo de *Software,* es un área que está en constante cambio tanto a nivel práctico como teórico. Por un lado, las nuevas tecnologías que a diario emergen, modifican de alguna manera los modelos que son propuestos desde la teoría; por el otro, muchos de los modelos de desarrollo propuestos se quedan en la teoría debido a una barrera existente entre la metodología y el uso de las herramientas de desarrollo. Los ejemplos de fracasos en cuanto a la adopción de una metodología en proyectos de desarrollo de *Software,* son bastante frecuentes en la práctica. Es posible que esto se deba a que las empresas tienen una necesidad de ordenar u organizar su proceso de desarrollo de alguna manera y recurren a alguna de las metodologías existentes. Sin embargo, este cambio en el proceso de desarrollo es bastante drástico e involucra absolutamente todas las etapas del desarrollo, desde las primeras entrevistas con el cliente, hasta la entrega final. Muchas empresas no logran adaptar de manera adecuada todos los procesos y el resultado resulta en demoras adicionales.

Sin embargo, el uso de una metodología adecuada ha probado ser un pilar para el desarrollo de un proyecto de construcción de *Software* (Moyo, Gonde, Soganile, Dzawo, & Madzima, 2013). Es aquí donde un gerente de proyectos novato tiende a hacerse la pregunta: ¿por qué todo es tan complicado cuando en el mercado existen múltiples metodologías de desarrollo de *Software* y es posible adaptarlas a nuestra cultura organizacional? Hay una respuesta simple y otra más complicada. La simple, dice que no es necesario adaptar la metodología solamente, sino que es necesario un cambio organizacional integral. La compleja, advierte que no toda metodología está diseñada para toda empresa, o mejor, que

no necesariamente cumple con las expectativas que tiene la compañía. Por esto, es necesario que al momento de adaptar una metodología, la organización previamente se tome el trabajo de conocer cuáles hay en el mercado, cuáles son las principales ventajas y desventajas, cómo es el proceso de implementación y si su alcance se encuentra alineado con el objetivo que busca, ya que toda metodología tiene un punto fuerte que la diferencia de las demás.

Con el fin de apoyar a las personas que deben tomar esta decisión, a continuación se ofrece una retrospectiva del panorama evolutivo de los modelos de *Software* y por qué los modelos de desarrollo ágil, particularmente Scrum, han logrado diferenciarse y ser una alternativa que aporta bondades y ventajas competitivas frente a otras existentes en el mercado.

2. Modelos de desarrollo de *Software*

Esta sección contiene un comparativo entre diferentes metodologías existentes y Scrum. Sin embargo, es importante empezar definiendo los conceptos de modelo y metodología, con el fin de forjar criterios claros en el desarrollo del artículo. Un modelo es un marco conceptual que define los componentes u objetivos que debe tener toda metodología que se quiera amoldar al proceso. Por su parte, la metodología se refiere como tal, a la adopción de un modelo y su especificación en cómo se implementa el concepto determinando al proceso; así la define: roles, entregables,

actividades y principales diferenciadores o reglas específicas. Como ejemplo, se puede tomar el modelo de desarrollo ágil, cuyo principal objetivo es el desarrollo iterativo e incremental, es decir, hacer fases muy cortas de desarrollo obteniendo un producto para revisión. De este modelo se pueden encontrar muchas metodologías como: eXtreme Programming (Stephens & Rosenberg, 2003), Scrum (Sutherland y Schwaber, 2010), Crystal Clear (Cockburn, 2004), entre otras. Además de las mencionadas, denominadas ágiles (Stellman & Greene, 2014), existe otro grupo que generalmente se conoce como metodologías clásicas o tradicionales (Pressman & Maxim, 2014). Algunos de los modelos que se pueden encontrar dentro de este grupo son: modelo en cascada, modelo basado de prototipos, modelo Incremental, modelo en espiral y el modelo de desarrollo ágil. Existen otros modelos pero son considerados adaptaciones de los anteriormente nombrados.

Por lo anterior es posible decir que existen dos grandes mundos en cuanto a metodologías se refiere: las tradicionales y las ágiles. Las tradicionales se caracterizan por el control a detalle, la subdivisión por etapas y la planeación, esto es, el seguir una guía clara partiendo del *Software* objetivo a desarrollar, las necesidades que lo componen, el cómo se va a desarrollar y posteriormente sí entrar a la etapa de desarrollo; por su composición, son metodologías que no se adaptan fácil a los cambios del alcance del *Software*. Por otro lado, las metodologías ágiles tienen como objetivo primario el hacer entregas rápidas, por ello, definen un lote de características relevantes que el *Software* debe cumplir y que además se alcancen a hacer en un marco determinado de tiempo -una a cuatro semanas-. Así, por medio de estas mini versiones ir tomando retroalimentación y evolucionando el producto; de esta manera la planeación está inmersa en las reuniones a realizar, pero no en todas las características que tendrá el *software* al final, esto por su adaptabilidad al cambio. Si se tuviera que poner un ejemplo, podría plantearse el que las

metodologías tradicionales se asimilan a un juego de parqués, en el cual se tiene en un inicio todas las fichas en la entrada y a continuación se siguen una serie de pasos con el fin de llevarlas al punto de salida -objetivo final-; si alguna de las fichas vuelve al punto de entrada tendrá que seguir nuevamente los pasos para llegar al final. Por el contrario, las metodologías ágiles se parecen más aun juego de ajedrez, donde constantemente hay que adaptarse a los cambios e ir cumpliendo pequeños objetivos en el transcurso del juego.

Es usual que en muchas empresas se continúe usando un modelo tradicional para desarrollar todo tipo de proyectos. Las metodologías tradicionales se basan en modelos lineales que consisten en etapas o fases como levantamiento de requerimientos -análisis-, diseño, desarrollo o implementación, verificación y puesta en producción. Las metodologías basadas en este modelo, han demostrado con el tiempo que tienen un gran inconveniente: los cambios en el alcance. Esto se debe al hecho que, para cumplir cada una de las etapas, se debe cerrar una puerta, es decir, si el proceso se encuentra en una etapa determinada, volver a una anterior implica costos adicionales que en ocasiones son bastante altos, dado que se hace necesario revisar nuevamente parte de lo que ya se ha hecho, pasar por la aprobación del cliente y modificar funcionalidades ya desarrolladas.

Una de las primeras alternativas que aparecieron frente a los modelos tradicionales, fue el llamado modelo basado en prototipos, el cual tenía como objetivo principal dar un acercamiento al cliente de lo que le sería entregado. Para lograr esto, una metodología basada en prototipos consiste en desarrollar prototipos a modo de formularios que simulan la funcionalidad del *Software*, sin implementar la lógica compleja del negocio, simplemente la parte visual. Este modelo tuvo gran acogida y es aún utilizado por la practicidad que brinda a la relación expectativa–realidad. Sin embargo,

la misma ventaja llega a ser su desventaja y es el no poder desarrollar el producto que se le había propuesto al usuario como un prototipo. Lo anterior, se debe a que la lógica del negocio puede ser mucho más compleja de lo que se prototipó o porque como resultado del primer prototipo, se encuentra que el cliente necesita una solución diferente. Ello implica hacer nuevamente otro prototipo y pasar por la aprobación del cliente. Adicionalmente, el hecho de ya tener aprobado uno involucra que cambios o mejoras al mismo son complicados de hacer cuando ya se está en la etapa de desarrollo.

Es importante mencionar que esta metodología fue aplicada durante dos años y que inicialmente fue muy buena ya que ofrecía al cliente la visión del producto terminado, pero a su vez, para un desarrollo particular de una ventana dinámica para carga de archivos, generó tiempos de desarrollos adicionales, ya que el prototipo que se había ofrecido y que para este caso fue fácil diagramar, requería una lógica adicional para ser implementada en el sistema que conllevaba más tiempo en la medida que debía generarse un nuevo componente para cumplir con la expectativa. De esto quedó una lección aprendida y es nunca pasar prototipos sin el aval de arquitectura.

El *Rational Unified Process* (RUP) (Kroll, Kruchten, & Booch, 2003), es una metodología tipo cascada iterativa -espiral- que trata de acercarse al cliente haciendo que la etapa de levantamiento de requerimientos se realice varias veces, lo que permite estar más abierto al cambio. Sin embargo, dos de las grandes desventajas del RUP es la disminución en el foco de atención en el levantamiento de requerimientos a medida que se itera; así mismo, su extensiva documentación, ya que para un cambio pequeño debían documentarse una serie de formatos basados en UML, que hacen que el proceso de documentación incremente los costos de desarrollo. Es así como se hace necesario replantear el uso de esta metodología

en algunos casos. Dentro de los documentos a elaborar se encuentran: diagramas UML como casos de uso, de secuencia, diagrama de componentes, entre muchos otros. Esto, sumado al hecho de que los tiempos de iteración que se definían -de dos meses o más- hacían pensar que no se tendría un segundo ciclo, complicando su resultado final.

La evolución de las metodologías continuó con las llamadas metodologías ágiles, una nueva filosofía cuyo principal objetivo es realizar entregas rápidas y adaptarse a los cambios. Dos de estas metodologías sobresalieron de las demás: *eXtreme Programming* (XP) y Scrum. Estas dieron un punto de quiebre en el tiempo, ya que ofrecían a los desarrolladores algo que no hacían las demás: manejar la documentación necesaria sin extenderse, apuntar a la satisfacción del usuario y adaptarse a los constantes cambios del cliente.

3. *Scrum* y su implementación

Scrum es una de las metodologías de desarrollo ágil de *Software* más reconocidas a nivel mundial, su concepción data de los años 80 en análisis realizados por Ikujiro Nonaka e Hirotaka Takeuchi, en el cual resaltaron el trabajo en equipo para el desarrollo de productos y la autonomía que estos deben tener (Takeuchi & Nonaka, 1986). Posteriormente, a principios de los años 90, fue retomado por Jeff Sutherland y Ken Schwaber, quienes formalizaron un marco de trabajo y unas reglas aplicadas particularmente al desarrollo de *software* de productos complejos (Schwaber & Sutherland, 2012). Este enfoque tomó mayor fuerza entre 2001 y 2010 cuando fueron generados elemento como el "Manifiesto ágil de desarrollo de *Software*" y la "Guía de Scrum". Dentro de este mismo período, se crearon varios programas y certificaciones con el fin de globalizar y

hacer de Scrum lo que es hoy en día, una de las metodologías más usadas, no solo en el área desarrollo de *Software*, sino en campos como fabricación, educación, entre otros.

La metodología fue llamada Scrum por el símil de trabajo en equipo que encontraron Nonaka y Takeuchi a una formación que lleva este nombre en *Rugby* y cuyo objetivo es en equipo, todos abrazados empujando hacia un mismo lado mientras el rival hace lo mismo en el sentido contrario, lograr la posesión del balón.

A pesar de que cualquier metodología puede parecer atractiva, y de hecho cada una tiene sus puntos positivos, es importante que desde la práctica esta se seleccione. El siguiente listado está compuesto por cinco ventajas que tiene el modelo Scrum sobre otro tipo de metodologías que se han logrado identificar, a través del uso de otras, en diferentes proyectos de *Software:*

- **Satisfacción del cliente:** el gran diferenciador de las metodologías ágiles, es que hacen al cliente parte del equipo de trabajo y lo comprometen con el resultado final. Esto es un gran cambio con respecto a metodologías tradicionales, en los que el cliente es una persona, o grupo de personas, con el que se realiza la tarea de levantar requerimientos o necesidades de tipo funcional y que posteriormente aprueba extensos documentos que no volverá a ver hasta que el producto se encuentre terminado. El uso de metodologías tradicionales en contraposición a un modelo de tipo ágil y en especial a Scrum, implica los siguientes riesgos:

  - Es posible que la funcionalidad que se había pactado en los documentos iniciales, cambie debido a diversos motivos, como políticas de la compañía o cambio de personal.

- Por razones administrativas, las personas que aprueban los documentos pueden ser diferentes a las que reciben el producto, por lo que se pueden tener discrepancias en cuanto a la entrega final.
- Llegar a largas discusiones con el cliente, porque la interpretación de las frases en el documento tienen diferentes expectativas para los diferentes interesados.
- Lograr que el cliente apruebe el sistema que se está entregando puede ser complicado al hacer el cliente o cambios gráficos.

SCRUM ayuda a solventar estos riesgos involucrando al cliente en el proceso de desarrollo. El cliente, en conjunto con el equipo de desarrollo, es quién define qué se hace y cuándo se hace. El equipo es el que se compromete con qué puede entregar en la duración del *sprint* -ciclo de desarrollo- e involucra al cliente en la inspección. Scrum hace parte integral de la inspección al cliente, permitiéndole realizar revisiones tempranas de los desarrollos y una revisión general del proceso, con el fin de que él mismo, con sus ideas, aporte al proceso, ayude a mejorar la sinergia del equipo y a que efectivamente se entregue lo que él espera.

- **Simplicidad**: los eventos manejados por Scrum están claramente identificados, indicando para cada uno: quienes participan, su objetivo, el tiempo que debe tomar y cuál es el resultado esperado. Lo cual en esencia facilita a los integrantes del equipo la adopción de la metodología.

- **Inspección:** uno de los componentes que resalta Scrum, es la inspección y por ello, tres de sus eventos están orientados a estos objetivos: la reunión diaria, la revisión del sprint y la retrospectiva de este último. Estos eventos permiten a la organización consolidar la metodología y detectar en cada equipo y en cada proceso, qué debe ser mejorado. Este componente es uno de los preferidos en las

organizaciones, ya que les permite ver qué tan bien se va adaptando la metodología a su cultura y si los beneficios prometidos se están evidenciando.

- **Adaptación:** la mejor parte de la metodología es la disposición que tienen al cambio las características del producto. Este es uno de los componentes que más la diferencia con el resto, ya que el cambio puede ser efectuado en cualquier momento, incluso dentro del desarrollo de la ejecución de las diferentes iteraciones o *Sprints* siempre y cuando no afecte la entrega pactada. Esta adaptación beneficia a la organización en la medida que aporta a la satisfacción del cliente y los ingresos por ajustes.

- **Trabajo en equipo:** algo particularmente interesante de Scrum es cómo logra la sinergia entre las personas que participan en el proceso, a tal punto que en cada iteración -ciclo de desarrollo-, el mismo equipo se adapta para mejorar. Esto también implica que cada individuo sea reconocido como parte esencial del equipo, por lo cual el impacto del cambio de una persona puede llegar a ser alto. Comparativamente, Scrum logra dar visibilidad al equipo de trabajo, ya que en metodologías tradicionales, las personas que lo integraban no tenían relación directa con el cliente, lo cual, en términos de reconocimiento, es importante y vale la pena destacarlo.

4. Comparación práctica con otras metodologías

En esta sección, se presenta una comparación entre metodologías de desarrollo de *Software*, resultado de la experiencia práctica de los autores en el campo de la construcción de software.

4.1 SCRUM vs metodologías tradicionales

Siendo Scrum una metodología ágil, el comparativo con las metodologías tradicionales parte de la base del comparativo que se realiza entre una tradicional y una ágil:

Las metodologías tradicionales se centran en la planeación de las actividades de principio a fin, subdividiéndolas en etapas, así: levantamiento de requerimientos, análisis, diseño, aprobación de diseños, construcción, pruebas y entrega. Scrum se basa en iteraciones cortas que entregan una parte del producto -incremento al producto- y no su completitud, para que a partir de esta el producto evolucione. Se presentan rígidas ante el cambio, ya que entre más próximo este al final del proceso, más difícil es realizar ajustes. Scrum está dispuesta al cambio ya que con iteraciones cortas, el mejorar o modificar una característica del sistema implica una labor tan sencilla, como priorizar el cambio e incluirlo en la iteración que corresponda.

Así mismo, las metodologías tradicionales manejan poca retroalimentación, pues si bien al inicio del proceso, es decir, en el levantamiento de requerimientos se está en constante contacto con el cliente, este se va perdiendo en las etapas de

diseño, construcción, pruebas y nuevamente se retoma en la entrega. Con Scrum, la retroalimentación es constante y no solo de las características del sistema sino también del proceso, de cómo se está desarrollando y cómo se puede mejorar. Estas metodologías centran su control en cómo va el desempeño con respecto a lo planeado, tanto en el proceso como en el software final. Scrum centra su control en qué tan satisfechos están los clientes con las versiones entregadas y cómo esto en cada iteración se puede mejorar.

Un caso práctico de las dificultades que se tienen con este tipo de metodología, se presentó en un proyecto para la Fiduciaria Bogotá en el cual la persona que levantó los requerimientos salió de la organización y fue reemplazada por otra que no estaba de acuerdo con algunas de las definiciones iniciales y que solicitaba cambiarlas. Teniendo en cuenta que ya se estaba en la fase de desarrollo, los cambios sugeridos tuvieron que negociarse con el cliente, unos se hicieron, otros no, y al final la insatisfacción del cliente era notoria. ¿Cómo pudo Scrum ayudar a que esto no pasará? Bueno, en cada ciclo de iteración existe una actividad llamada *Sprint Review* que busca invitar a los interesado a que den sus opiniones acerca del software y su avance; en este punto, más personas del área hubieran participado y ante varios puntos de vista y diferentes aprobaciones, el usuario que recibía el sistema debía entender que era una decisión en conjunto y no de una sola persona.

4.2 Scrum vs otras metodologías ágiles

Scrum, enmarcada como metodología ágil, cumple con muchos de los principios que tienen estas: entrega de productos funcionales tempranos, constante retroalimentación en el equipo de trabajo, adaptabilidad a los cambios y trabajo en equipo con el cliente. Sin embargo, ya entrando al detalle, tiene unos diferenciadores importantes que vale la pena destacar:

- **El tiempo:** en metodologías como *eXtreme Programming* (XP), el marco de tiempo de iteración es menor, se habla de una o dos semanas. Scrum trabaja con marcos de tiempo de dos a cuatro semanas.
- **El proceso**: Scrum en sí mismo, no define las herramientas a utilizar para implementar su proceso, es la organización o el scrum master quién da el lineamiento de qué se debe usar. Otras metodologías de desarrollo ágil enmarcan las herramientas, definiéndolas como parte del estándar para su uso. Por ejemplo: desarrollo orientado a pruebas en XP.
- **Los cambios:** ya dentro de la iteración Scrum, se sugiere no hacer cambios a los compromiso adquiridos, otras metodologías de desarrollo ágil permiten hacer cambios dentro del proceso de desarrollo de la iteración, es decir, son más flexibles.
- **El orden:** en Scrum se permite que el equipo con base en la prioridad definida por el cliente sea quién decida en qué se puede comprometer para cada iteración en cuanto a desarrollo se refiere, en XP el equipo debe seguir el lineamiento dado por el cliente.
- **La retroalimentación:** Scrum sugiere retroalimentación al finalizar cada sprint, en otras metodologías como XP se sugieren retroalimentaciones tempranas a medida que se desarrolla la entrega.

Scrum, particularmente con uno de nuestros clientes principales –EPM- terminó ofreciendo un gran beneficio y fue la satisfacción del cliente con el producto terminado. Antes de Scrum se llevaban los documentos requerimiento, y por medio de correos iban y venían hasta lograr un entendimiento, se hacían las debidas reuniones de socialización y prototipos cuando era necesario, posteriormente se desarrollaba, se realizaban pruebas y se hacia la entrega, y era en este último punto donde nos percatábamos en conjunto que la definición había quedado incompleta o que se requerían mejoras al desarrollo; incluso, que habían frases que se habían mal

interpretado. Todo esto redundaba en un relación caótica con el cliente ya que el siguiente paso era la no aceptación del producto hasta su completitud y se tenía entonces que entrar a negociar estos ítems y terminar asumiendo trabajos no estimados en aras de la satisfacción final y de la buena relación.

Con Scrum se cambió el modo en el que el cliente percibía el trabajo, pues siendo parte del equipo, entendió el trabajo que se desarrollaba; acompañaba el proceso y realizaba revisiones tempranas, notando una mejora en las entregas en la medida que llegaban con muy buena calidad y justo lo que él quería. Lo anterior, por el simple hecho de cambiar el rol, en donde dejó de ser un cliente que espera un producto de un proveedor, para pasar a ser un cliente que junto con el proveedor, crean un equipo para entregar a la organización lo que necesita. Se debe reconocer que en ocasiones se dieron sobrecostos para el cliente por mejoras que se incluían en el proceso de desarrollo, pero la evaluación final siempre mostró que valió la pena por el producto recibido.

4.3 Limitaciones de SCRUM

Como toda metodología, Scrum a pesar de las bondades que presenta, también tiene limitaciones que deben tenerse en cuenta a la hora de ser implementado:

- **Complejidad en su implementación**: para equipos que vienen del desarrollo tradicional enfrentarse a ideas tales como ser auto-gestionados, estar abierto a cambios que pida el cliente, o estar en contacto directo con el cliente, puede ser todo un reto y requiere ejercicios previos antes de ponerse en práctica.
- **Tiempo por parte del cliente:** esta metodología requiere tiempo considerable por parte del cliente y contacto permanente con el equipo, por lo cual puede que no en

todos los escenarios se cuente con la disponibilidad de trabajar de esta forma.

- **Contratos costo y alcance definido:** aunque es posible trabajar con Scrum en este tipo de contratos, se pierde algo básico de la esencia y es la adaptabilidad, el poder cambiar o mejorar características del sistema a medida que evoluciona el producto. Lo anterior. Considerando que se tiene una fecha objetivo para cumplir con las características bajo las cuales se hizo la contratación.
- **Documentación requerida:** parte del objetivo de las metodología ágiles, es documentar lo estrictamente necesario, pero ¿Qué pasa cuando el cliente es el que exige documentación exhaustiva y prototipos detallados previos al desarrollo? En estos escenarios Scrum no luce tan bien, pues atenta contra la base fundamental que es el agilísimo.
- *Stress*: este es un fenómeno que se da no solo hablando de Scrum sino en general de cualquier metodología de desarrollo ágil, y es que el equipo, en aras de cumplir con las entregas en ciclos cortos que se repiten indefinidamente, cae en una etapa de *stress*, ya que los miembros siempre están pensando en compromisos, tiempo y estándares de desarrollo.
- **Proyecto de gran envergadura:** Scrum está pensado para un equipo de hasta nueve personas, si más de esto es requerido, se debe pensar en varios equipos de Scrum, lo cual a su vez, implicaría una coordinación y un seguimiento que puede no ser fácilmente aplicable en el proyecto.
- **Ingenieros junior:** debido al aprendizaje y sinergia que logra el equipo, la inclusión de ingenieros con menos experiencia o sin experiencia en la metodología, implica un acompañamiento muy de cerca que puede llegar a comprometer los tiempos que se tienen para el desarrollo del producto.

5. Conclusiones

Este artículo presenta diferentes modelos de desarrollo de *Software* comparados con la metodología llamada Scrum. Cada una de estas metodologías tiene sus ventajas a la hora de afrontar el desarrollo de proyectos de *Software*. Más allá de las diversas discusiones que se dan por saber cuál es mejor, una de las ventajas que tiene Scrum, en términos de implementación, es que es comparativamente más sencillo que otras metodologías, ya que no exige la adopción de prácticas generales de ingeniería que otras sí referencian. Esto no supone una falencia, sino que deja espacios para que la misma organización sea quién defina que técnicas utilizar en ciertas etapas del desarrollo -por ejemplo: tipos de prueba a realizar-, lo cual se considera que da un mayor margen de adaptabilidad a la cultura organizacional.

Se debe considerar que la adopción o implementación de una nueva metodología siempre presenta inconvenientes en la medida que es un cambio en los procesos de una organización y teniendo en cuenta el elemento humano, no toda persona tiene la disposición de adaptarse. Por esta razón, este artículo resalta factores de por qué Scrum puede ser una de las mejores opciones al momento de seleccionar una metodología de desarrollo. Esto no implica que en su implementación se puedan presentar inconvenientes de diferente índole, y que tal como lo plantea la metodología, deben ser solventados en equipo, bajo la guía de un experto (*Scrum Master*).

Se considera que los beneficios que puede traer el implementar Scrum para una compañía pueden ser múltiples, partiendo de una mejor relación con el cliente, entendimiento del objetivo final, ingresos acordes con las labores realizadas.

Por lo tanto, es una metodología que vale la pena evaluar. El contar con una metodología que base sus principios en la transparencia, los individuos, la adaptación y la importancia del cliente como conocedor del objetivo final, es un primer paso que puede conducir a una organización a una mejora del clima laboral y el reconocimiento en el medio.

Se espera que las razones indicadas en este artículo justifiquen y animen al lector a probar una metodología que realmente ofrece un valor agregado como lo es Scrum.

6. Referencias bibliográficas

Cockburn, A. (2004). *Crystal Clear: A Human-Powered Methodology for Small Teams:* A Human-Powered Methodology for Small Teams. New York: Addison Wesley.

Kroll, P., Kruchten, P., & Booch, G. (2003). *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*: A Practitioner's Guide to the RUP. New York: Addison-Wesley.

Moyo, B., Gonde, P., Soganile, N., Dzawo, G., & Madzima, K. (2013). *Empirical evaluation of software development methodology selection consistency: A case study using Analytical Hierarchy Process. Proceedings of the International Conference on Software Engineering Research and Practice* (SERP), (págs. 1-7). Athens.

Pressman, R., & Maxim, B. (2014). *Software Engineering: A Practitioner's Approach (8ed.)*. New York: McGraw-Hill Education.

Rubin, K. S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. New York: Pearson Education.

Schwaber, K., & Sutherland, J. (2012). *Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitors In the Dust. Wiley.*

Stellman, A., & Greene, J. (2014). *Learning Agile: Understanding Scrum, XP, Lean, and Kanban*. New York: O'Reilly.

Stephens, M., & Rosenberg, D. (2003). *Extreme Programming Refactored: The Case Against* XP. New York: Apress.

Takeuchi, H., & Nonaka, I. (1986). Takeuchi, Hirotaka, and Ikujiro Nonaka. "*The New New Product Development Game. Harvard Business Review*, 64(1), 137-146.