KnE Engineering

SIIPRIN-CITEGC
Ibero-American Symposium on Computer Programming jointly held with the International Congress on Technology Education and Knowledge Management
Volume 2020

Knowledge E
enriching | engaging | empowering

Conference Paper

MEAC: An experience to keep the production line active in the software development process

MEAC: Una experiencia para mantener la línea de producción activa en el proceso de desarrollo de software

Luis Caiza[1], Doris Chicaiza[1], Rolando P. Reyes Ch.[1,2], Franklin Montaluisa[1]

Maestría en Ingeniería de Software
[1]Universidad de las Fuerzas Armadas-ESPE, Latacunga, Ecuador
[2]Universidad de las Fuerzas Armadas-ESPE, Sangolqui, Ecuador

Abstract

Software development is not an easy process. Proof of this is the existence of numerous methodological proposals that have proven to be practical and efficient in a large number of projects. However, these methodologies have also presented problems in aspects of their application. To such an extent that they often need to be adapted to the realities of each project or company in order to improve the results obtained in their application. The aim of this article is to tell a real experience in a small development company in which several methodologies have been applied over the years, highlighting *Scrum* as the one that has brought better results. However, by applying it specifically to small projects, with few resources, limited times and changing requirements, the need to also adapt to this methodology for this type of project was born. We have called this adaptation *MEAC*, acronym that refers to an Empirical Method of Continuous Support whose purpose is to avoid time cuts in the generation and development of the software product, as well as to maintain the active production line of the software team active.

Resumen: El desarrollo de software no es un proceso fácil. Prueba de ello, es la existencia de numerosas propuestas metodológicas que han demostrado ser prácticas y eficientes en un gran número de proyectos. No obstante, estas metodologías también han presentado problemas en aspectos de su aplicación. A tal punto que, muchas de las veces se necesitan adaptarlas a realidades propias de cada proyecto o empresa con el fin de mejorar los procesos de desarrollo de software y obtener un producto de calidad. El objetivo de este artículo es contar una experiencia real en una empresa pequeña de desarrollo en la cual se ha aplicado varias metodologías al largo de los años, destacándose *Scrum* como la que mejores resultados ha traído. Sin embargo, al aplicarla específicamente en proyectos pequeños, con pocos recursos, tiempos limitados y requisitos cambiantes, ha nacido la necesidad de también adaptar a esta metodología para este tipo de proyectos. A esta adaptación la hemos llamado *MEAC*, siglas que se refieren a un Método Empírico de Apoyo Continuo cuya finalidad es evitar cortes de tiempo en la generación y desarrollo del producto software, así como mantener activa la línea de producción activa del equipo de software.

Corresponding Author:
Luis Caiza
lhcaiza@espe.edu.ec

Received: 24 December 2019
Accepted: 2 January 2020
Published: 8 January 2020

Publishing services provided by Knowledge E

Selection and Peer-review under the responsibility of the SIIPRIN-CITEGC Conference Committee.

Keywords: Production line; Scrum adaptation; Development Times; Empirical method of Continuous support

Palabras claves: Línea de producción; Adaptación de *Scrum*; Tiempos de Desarrollo; Método empírico de apoyo Continuo

1. Introducción

Una metodología de software es un marco de trabajo que permite desarrollar software eficientemente y con calidad. Por ello, todo proyecto de software se debe planificar, organizar y desarrollar con habilidad y conocimiento. Las metodologías de software son utilizadas de diferente manera dependiendo del tamaño y necesidades de las empresas [1]. En esta evolución las pequeñas empresas de desarrollo de software generan un proceso de madurez, donde común mente se pasa de aplicar procesos empíricos a metodologías tradicionales [2] y finalmente a las metodologías ágiles [3]. En cierta manera, estos cambios permiten mejorar los procesos de este tipo de empresas. Esto es evidencia que la tecnología, así como las metodologías de desarrollo de software avanzan, de tal manera que las empresas sobre todo de tecnología se ven obligadas a innovar junto con estos avances, ya que estar tecnológica y metodológicamente actualizados es un punto importante de competitividad frente al cliente.

En la actualidad el proceso desarrollo de software aún sigue siendo altamente empírico sobre todo en empresas pequeñas [4], y medianas donde el empirismo está presente desde el levantamiento de requisitos hasta el desarrollo, a pesar de la existencia de las metodologías, métodos y buenas prácticas, existen limitaciones propias del tamaño de las mismas, esto influye también en las metodologías aplicadas ya que muchas veces estas necesitan adaptaciones para ser usadas bajo las necesidades propias de las empresas o simplemente las metodologías existentes enfocan a empresas grandes. Un ejemplo de este problema es la cantidad de recursos disponibles para aplicar una metodología pura, ya que al no contar con los recursos suficientes los integrantes de las empresas pequeñas se ven obligados a cumplir actividades adicionales no propias de sus funciones.

Como consecuencia de esta sobrecarga de actividades, existe pérdida de tiempo en el proceso de desarrollo, retrasos en la entrega de productos, incertidumbre de roles, tareas y responsabilidades, lo cual impiden mantener una línea de producción activa

de desarrollo, así también provocando desfases en los cronogramas, costos elevados, insatisfacción del cliente con el producto terminado entre otros.

El objetivo de este artículo es describir las experiencias de un equipo de desarrollo de software al pasar de usar metodologías tradicionales a usar la metodología *Scrum*, los aciertos y equivocaciones al aplicarla, el análisis de los problemas presentados en proyectos pequeños y las adaptaciones realizadas a la misma, generaron así la propuesta del Método de Apoyo Continuo (*MEAC*) como una alternativa para solventar los problemas encontrados. Adicional, se muestran estadísticas en un análisis comparativo sencillo al usar *Scrum* sin adaptaciones en un proyecto real frente a *MEAC*. De esta manera se explica el comportamiento de cada uno de ellos (*Scrum* y *MEAC*) en proyectos pequeños.

En las siguientes secciones explicaremos cómo se usó *Scrum* y *MEAC* en los proyectos dentro de la empresa. La sección 2 describe los trabajos relacionados; la sección 3 narra de manera breve las experiencias de la empresa al aplicar metodologías de desarrollo de software. Seguidamente, la sección 4 que detalla el portafolio de problemas identificados al aplicar *Scrum* en pequeños proyectos, la sección 5 describe las causas de problemas identificados en la sección 4, la sección 6 describe el Método de Apoyo Continuo aplicado a proyectos pequeños de software, la sección 7 se habla de la metodología de investigación aplicada al caso de estudio, la sección 8 describe los resultados de la aplicación de *MEAC* vs *Scrum* al desarrollar software, la sección 9 propone las conclusiones y discusión respecto al caso de estudio. Finalmente, la sección 10 propone trabajos futuros como y donde se aplicar *MEAC*.

2. Trabajos Relacionados

En la literatura, existen varios autores [4, 5]que han creado diferentes metodologías para desarrollar software [6]. Han creado ágiles [7], incrementales, tradicionales [8], híbridas, por etapas, etc. En la mayoría de los procesos de desarrollo de software, las metodologías han permitido tener orden, disciplina y una excelente gestión para que las tareas sean eficientes mejorando los procesos de desarrollo [9].

Al respecto, es importante señalar también el trabajo realizado por Medina y López [10], que proponen el método Winner para desarrollar software, compuesto de cuatro etapas como comprensión y conocimiento, planteamiento, formalización y complementos, cada uno de ellos poseen procesos, actividades y tareas, como artefacto obtiene tablas de resultado entre ella son tabla de descripción de objetivos, tabla de tipos de usuario, tabla de requisitos, etc., lo cual proporciona agilidad en el desarrollo, entrega

continua, control de calidad en el producto final por lo tanto es manejable por equipos de desarrollo pequeños o grandes. Por otro lado, Kaur et al [11], explican que una metodología ágil aplica la flexibilidad y respuestas rápidas ante los cambios en procesos de desarrollo, restablecen el equilibrio en el proceso de intercambio de información para mejorar el acoplamiento de la comunicación entre los equipos del proyecto y los usuarios lo que da una pauta para entender qué es fundamental la comunicación clara de los roles para una correcta organización del proceso de desarrollo.

Según Romano y Delgado [12], presentan la adopción de una solución ágil, esta elección lo hicieron porque la empresa tiene una serie de problemas de desarrollo en proyectos de pequeño y mediano tamaño, cuenta con pocos empleados y, en general, cada uno desempeñando más de un papel en la organización, los desarrolladores a menudo son interrumpidos para resolver problemas relacionados con otros proyectos. Por ello se apoyaron en *Scrum*, adaptaron las características, buenas prácticas con un ciclo de vida compuesto por infraestructura establecida, entrenamiento en equipo, despliegue ágil de *Scrum*, despliegue de refinación, lo cual permitió tolerar las interrupciones en proceso de desarrollo y entregar el producto a tiempo.

Otros autores clave en el desarrollo de metodologías ágiles con *Scrum*, son Srivastava et al. [13], exponen un marco de trabajo diseñado para lograr la colaboración eficaz de equipos en proyectos, que emplea un conjunto de reglas y artefactos para definir roles que generan la estructura necesaria para su correcto funcionamiento.

La principal novedad del trabajo que aquí se presenta frente a otros es la propuesta de mantener siempre la línea de producción de software activa, permitiendo que los roles que giran alrededor del desarrollador trabajen para garantizar este proceso. Si bien los trabajos mencionados, asemejan las características de los procesos a resolver con la propuesta del paper, está se enfoca en resolver los problemas específicas del caso de estudio presentado.

3. Experiencia -- Transición a Scrum

3.1. Historia de la empresa

Al ser una empresa pequeña de desarrollo de software con más de 15 años en la industria enfocada a crear productos a medida y productos propios de comercialización. Como empresa se ha buscado la manera de mejorar los procesos de producción de software a lo largo del tiempo. Es así que se pasó de aplicar el empirismo a aplicar la metodología tradicional orientada a objetos. Esto permitió mejorar procesos internos

de gestión y desarrollo. Sin embargo, las nuevas metodologías de desarrollo obligaron a la empresa a evolucionar para estar a la par de las necesidades del cliente por ello para el año 2017, se tomó la decisión de evolucionar y aplicar la metodología ágil *Scrum* [14]como marco de trabajo a la mayoría de los procesos de desarrollo de la empresa, con varios cambios dentro de la organización, las cuales incluían:

- Organización del equipo interno basado en roles de *Scrum* y Capacitación continua.
- Distribución de funcionalidades y responsabilidades en base a los roles.
- Creación Formal del área de gestión de proyectos (Product Owner)
- Otros ajustes como reuniones, estimaciones, ejecución de proyectos.

3.2. Primeros pasos con Scrum

Al principio, el cambio a *SCRUM* fue bastante fácil, debido a que todos los miembros de la empresa estaban comprometidos en aplicar los procesos de la metodología. Esta metodología prometía resolver varios problemas que se venían acarreando desde varios años en algunos procesos de gestión y desarrollo. Para ello se redistribuyo la organización a roles recomendados por *Scrum* [15], véase Tabla 1:

TABLE 1: Áreas, Roles y Actividades.

Área	Rol Definido	Rol Scrum	Actividades
Gerencia	Gerente	StakeHolder Interno	Establece relaciones con los clientes a nivel político. Generar nuevas oportunidades de venta. Apertura de mercados y otros procesos gerenciales
Gestión de proyectos	Gestor	Product Owner	Control de proyecto cronogramas, precios, recursos, tiempos y compromisos.
Equipo Técnico	Equipo de Desarrollo	Scrum Master	Son desarrolladores con más experiencia, y dirigen equipos.
		Equipo Scrum	Son los desarrolladores, quienes generan el producto.

En los primeros proyectos que se aplicó *Scrum* fueron medianos [13], se cumplió con los objetivos establecidos y se pudo emplear todos los roles recomendados por *Scrum*, en donde se identificó el éxito total en la migración a la nueva metodología respecto a la antigua metodología de desarrollo, y se obtuvo mejores resultados por cada proyecto en: *la distribución del trabajo interno, el seguimiento del estado real del proyecto, soportar cambios sobre los requerimientos, casi ningún o muy pocos desfases en los cronogramas, mejor comunicación con los equipos de los clientes,*

retrospectivas exitosas en el desarrollo, mejor percepción de cumplimiento con los clientes.

4. Portafolio de problemas al aplicar Scrum

En el transcurso del tiempo y al realizar inspecciones en proyectos pequeños de software de la empresa, se halló que estos no se ha adaptan completamente a *Scrum*, y como consecuencia se generan desfases en los tiempos de entregas, malestar en los clientes y equipo de desarrollo, llegando a identificar una cartera de problemas los cuales se detallarán a continuación.

4.1. Equipos pequeños y mala organización

Se ha identificado que el 70% de los proyectos de la empresa son pequeños, con un tiempo de desarrollo aproximado de 1 a 2 meses, y para poder establecer al proyecto como rentable se debía asignar una o dos personas en desarrollo y una persona en gestión de proyectos, este factor impedía aplicar los cuatro roles recomendados por *Scrum* [14] que sugiere tener al menos los 4 personas, 1 por cada rol (StakeHolder, Product Owner, *Scrum* Master y Equipo *Scrum*). Prácticamente los integrantes del equipo de desarrollo adoptan roles y responsabilidades que no corresponden dentro de sus actividades.

4.2. Recursos limitados

Otro de los problemas identificados en el proceso de desarrollo de software, es los recursos limitados de la empresa, debido a una recesión le impedía contratar personal para rellenar roles faltantes. El objetivo principal era resolver los problemas utilizando los recursos disponibles, esto obligó a pensar cada uno de los integrantes del equipo sin provocar a la vez un caos interno. Autores como Rodríguez et al. Propone que la optimización y organización de recursos, se centran en mejorar el rendimiento, la utilidad, reducción de los costos y del tiempo de desarrollo dependiendo de las necesidades del proyecto[16].

4.3. Tiempos Perdidos

Dentro del portafolio de problemas, el *Equipo Scrum* identificó y analizó las razones de los tiempos perdidos en el proceso de desarrollo, y los factores encontrados fueron varios, como asistir a reuniones de presentación, reuniones de negociación, pruebas con el cliente, instalaciones en ambientes y documentación, el tiempo se acortaba considerablemente. Estas actividades adicionales a su rol impedían generar software de manera continua, además no se podía cumplir con los tiempos establecidos, generando malestar entre el equipo de desarrollo, evidenciando la mala organización de los roles, Miramontes et al. [17],propone el uso estrategias para aligerar procesos de desarrollo y evitar tiempos perdidos, las estrategias son actividades propias al equipo, aplicación de prácticas agiles y métodos propios de las necesidades de cada proyecto.

4.4. Cliente mal organizado

En la metodología *Scrum* se considera al cliente como la parte fundamental del proyecto, pero al no tener claro su rol dentro del proyecto el cliente se comunicaba con cualquiera de los roles intentando pedir información, organizar reuniones, cambios y quejas lo que provocaba un caos en el equipo interno ya que no se tenía una línea de comunicación clara.

4.5. Otros Factores

La presión que el cliente ejercía sobre el equipo de desarrollo, las entregas a pruebas semanales, presentaciones en cronogramas no definidos y publicaciones en fechas no establecidas provocaron que varias de las horas programadas de generación del producto se disminuyan considerablemente.

5. Causas de los problemas encontrados

Para el estudio de caso se decidió analizar uno de los proyectos con más inconvenientes. Al proyecto se le llamara ``Proyecto 1", este padecía todos los problemas detallados en el portafolio de problemas, el análisis permitió identificar las verdaderas causas de los problemas al aplicar *Scrum* en equipos pequeños, a continuación, se detalla:

5.1. Líneas de Producción Cortadas

Al analizar la cantidad de horas invertidas en el proceso de desarrollo y comparar con las horas trabajadas por el *equipo scrum*, se identifican que las líneas de producción se cortaban de forma regular. Es decir que estos cortes, son tiempos utilizados en otras actividades como reuniones, pasos a ambientes, documentación y pruebas que no pertenecen al proceso de desarrollo propiamente dicho, lo cual impedía que puedan generar el producto de forma continua, esto fue provocado porque el *equipo scrum* cumplía con funciones extras del *Scrum Master* y *Product Owner*, lo que rompía con la recomendación de la metodología [18].

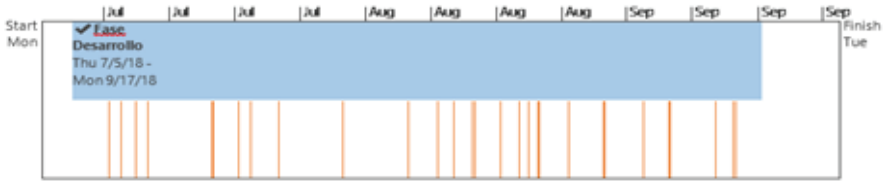

Figure 1: Línea del tiempo de la fase de desarrollo del ``Proyecto 1''.

En la Figura 1, se puede ver en color celeste la línea de producción en la fase de desarrollo del proyecto y en color rojo cada corte de tiempo al proceso (reuniones, instalaciones, pruebas, documentación) donde claramente se denota que existen cortes continuos a la línea de producción.

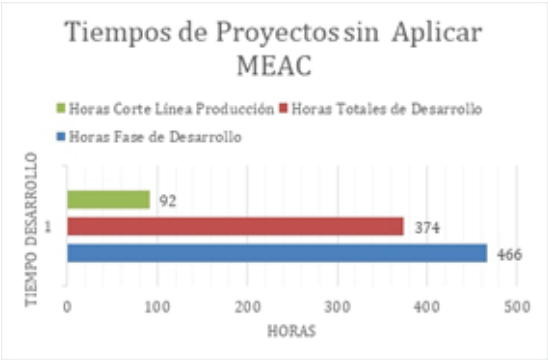

Figure 2: Tiempos de duración del Proyecto.

Como se puede observar en la figura No2, Se establece los tiempos de desarrollo para un proyecto, en este caso ``Proyecto 1'', en donde se especifica la fase y horas de desarrollo. Con un tiempo establecido de 466 horas en la fase de desarrollo total, siendo el 100% del tiempo invertido en el ``Proyecto 1'', 374 horas de desarrollo específicamente equivalente al 80,20% de desarrollo, y 92 horas de corte de línea de producción que representa al 19,74% del tiempo perdido.

5.2. Equipo Scrum cumpliendo otras responsabilidades

En el caso del equipo *Scrum*, se analizó las responsabilidades que realiza cada persona en el proceso de desarrollo. Se observó que la mayoría de las actividades recaen sobre el desarrollador. Este se encarga de organizar su cronograma, documentar, comunicar, asistir a reuniones con el cliente, reuniones técnicas, realizar publicaciones en ambientes, e incluso negociar requisitos y cronogramas del proyecto, por lo que el desarrollador cae en una sobrecarga de actividades y descontrol de su proceso de generación del producto, lo que coincide con lo mencionado por G. P. Tomaselli et al. [19] que a pesar que *Scrum* es ampliamente utilizado se caracteriza la ausencia de guías claras u homogéneas de cómo implementar los procesos de producción de software.

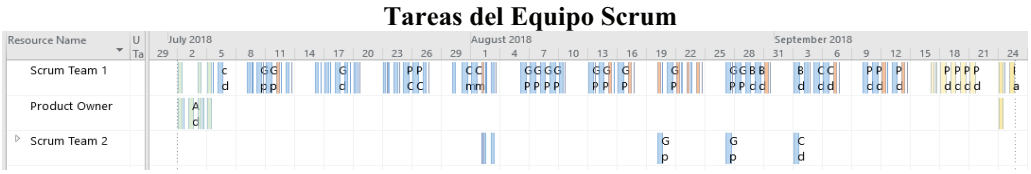

Figure 3: Tareas del Equipo Scrum.

En la Figura 3, se muestra la mayoría de los procesos eran realizados por el Desarrollador mientras que el *Product Owner* sólo participaba de algunas tareas y no de forma continua. Con este análisis se entiende que los roles de *Producto Owner y Scrum master* no cumplieron con su trabajo. Se propuso identificar las tareas de *Scrum* master, pero para este proyecto no fue asignado, solamente se designa un desarrollador y otro adicional que participa en ciertas tareas del desarrollo.

5.3. Cliente cambiante (o requerimientos volátiles)

Las metodologías ágiles hacen al cliente parte del equipo de trabajo y lo comprometen con el resultado final [20], sin embargo este pedía cambios de requerimientos en reuniones que no estaban planificadas, ni esperaban la finalización de un sprint para solicitarlo, se planteó la pregunta ¿Por qué lo hace? y este definió que lo hacía ya que quería aprovechar el tiempo que tienen en las reuniones, además que no existía una persona quien organice las reuniones de los temas referentes al proyecto. Otro factor recae en que el desarrollador es miembro activo de las reuniones con el cliente mientras que el *Product Owner* tiene poca participación en el proceso.

6. Método Empírico de Apoyo Continuo (MEAC) en el proceso de desarrollo de software

En base al análisis detallado en la sección 4 y sección 5, en la cual se determinó los problemas y causas que ocasionan desfases en las fechas de entrega y cuáles son las consecuencias en un proyecto. En definitiva, se puede concluir que la empresa tiene una mala organización de los integrantes de los proyectos, sus roles y responsabilidades al ser aplicados en proyectos pequeños lo que provoca que la línea de producción generada por el *Equipo Scrum* tenga demasiados cortes, con un tiempo perdido de hasta un 20% del tiempo global del desarrollo.

Por ello nace *MEAC* Método Empírico de Apoyo Continuo con el fin de garantizar que factores internos y externo afecten lo menos posible al rol generador del producto el ``Equipo *Scrum*'', de tal manera que se prioricen las tareas de estos sobre cualesquiera otras tareas, y haciendo que los roles que giran a su alrededor brinden un apoyo continuo al equipo *Scrum*. A continuación, describiremos el *MEAC* y su aplicación en un nuevo proyecto.

6.1. Organizando el equipo Interno en MEAC

A diferencia de *Scrum*, se plantea que *MEAC* establezca las nuevas responsabilidades de los roles en el proceso de desarrollo de software dependiendo de la cantidad de personas asignadas a un proyecto, véase Tabla 2:

Con estos ajustes al proceso se pretende siempre tener una línea de producción activa donde los roles alrededor del equipo *Scrum* son un apoyo vital para el cumplimiento de los objetivos de cada proyecto, ahora el enfoque empresarial es ``*Siempre Mantener La Línea De Producción Activa*''.

6.2. Roles y portafolio de problemas.

A continuación, mostramos los roles fusionados y los problemas a los cuales van a atacar cada uno de ellos, véase Tabla 3:

7. Metodología

Para llevar a cabo este estudio se decidió usar la metodología de investigación cualitativa [22]y la técnica de observación [23]ya que era necesario determinar las causas

TABLE 2

Rol	Fusión de Roles	Características	Responsabilidades
StakeHolder	**StakeHolder**	Generar la idea y apertura del proyecto. Seguimiento y verificación del resultado del proyecto entregado.	General la visión del proyecto.
Product Owner	**Product Owner**	Product Owneer toma las responsabilidades del Scrum Master. Medio de comunicación entre el cliente y el desarrollador. Todo factor de negocio será	Generar épicas. Generar historias de usuario Generar estimaciones Generar backlog y sprints Generar cronogramas Reuniones diarias
Scrum Master		tratado por este rol. Es su responsabilidad y solo suya Mantener la línea de producción activa.	Seguimientos Negociones Comunicaciones Retro Alimentaciones Presentaciones y planificaciones.
Equipo Scrum	**Equipo Scrum**	Persona Técnica y Analista. Informar novedades que impidan generar el producto. Auto organización de su trabajo. Responsabilidades definidas Generar el producto continuamente	Desarrollar el Producto de forma Continua. Comunicación continua con el producto Owner. Generar y verificar Insumos Ayudar en Cotización Realizar Instalaciones o publicaciones. Reuniones Diarias

TABLE 3: Roles y portafolio de problemas.

Rol	Problema	Solución
StakeHolder	Cliente mal organizado	El cliente se comunicará únicamente con el *Product Owner* este será el único canal que tome sus inquietudes, problemas y quejas y las resolverá, el cliente tiene claro cuál es la fuente oficial de comunicación.
Product Owner	Equipos pequeños y mala organización	El rol tiene claro cuáles son sus responsabilidades y es su responsabilidad organizar tanto al equipo interno y externo
	Recursos Limitados	Fusionar el Rol de *Product Owner con* el *Scrum Master* permitirá usar solo un recurso en vez de dos para ello el integrante que ocupe este rol debe tener conocimientos tanto técnico como de gestión de tal manera que pueda cumplir con las responsabilidades de los dos roles.
	Otros Factores	Cualquier imprevisto que impida que el Equipo Scrum genere el producto será responsabilidad del *Product Owner* resolverlo.
Equipo Scrum	Tiempos Perdidos	El desarrollador se encargará de generar el producto como su prioridad, dejara de cumplir tareas que le impidan hacerlo, en caso de existir otros problemas se las comunicará al *Produc Owner* y será ese rol quien se encargue de resolverlo

que provocan no poder cumplir con los tiempos comprometidos para la ejecución de

los proyectos pequeños, para ello se le comunicó a cada integrante del proyecto que se realizaría la observación de los factores que intervienen en la ejecución del mismo y además se mediría el tiempo que le toma en la ejecución de cada tarea, siempre especificando que esta medición no sería objeto de ninguna sanción o calificación de productividad, sino que servirían para mejorar el proceso actual de desarrollo y que sería beneficioso para todos.

Una vez obtenidos los datos estos fueron analizados estadísticamente, categorizando cuales tareas eran propias del desarrollo y cuáles no, de tal manera que se evidenció la cantidad de tiempo invertido en cada una de ellas. Además, se llevó el cronograma de ejecución de las mismas lo que permitió entender en qué periodo de tiempo se daban las tareas ajenas al desarrollo, por parte de quienes eran ejecutadas, los roles y responsabilidades.

8. Resultados

Según los ajustes antes mencionados, en la Tabla No.2 y Tabla No.3 se aplicó a un nuevo proyecto de nombre ``Proyecto 2'', el cual tenía como características ser pequeño, con 4 semanas de duración en todas sus fases, con un tiempo aproximado de 2 semanas dedicadas solo al desarrollo, un caso perfecto para poner a prueba los puntos planteados.

8.1. Tiempos de desarrollo con MEAC

Al aplicar el método de apoyo continuo, se observó un mejor desempeño del desarrollador ya que no realizaba tareas externas a su proceso lo que provocó que los cortes de tiempo fueran los absolutamente necesarios, mejorando así el tiempo de desarrollo, el cumplimiento de sus responsabilidades y los objetivos planteados en el proyecto.

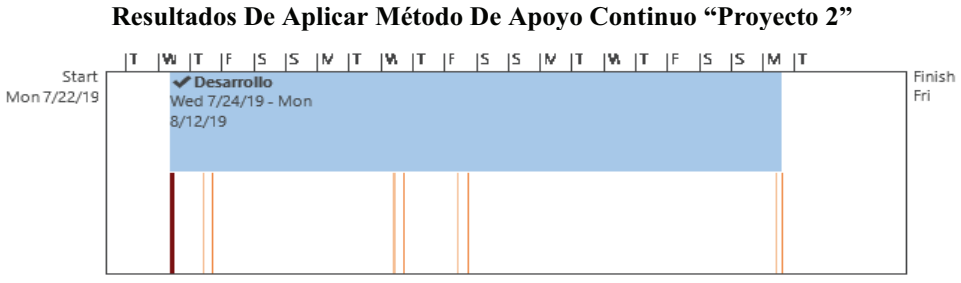

Figure 4: Resultados De Aplicar Método De Apoyo Continuo ``Proyecto 2''.

En la figura 4, se puede observar que los cortes de tiempo disminuyeron de manera considerable, y las tareas fueron realizadas en fecha específicas.

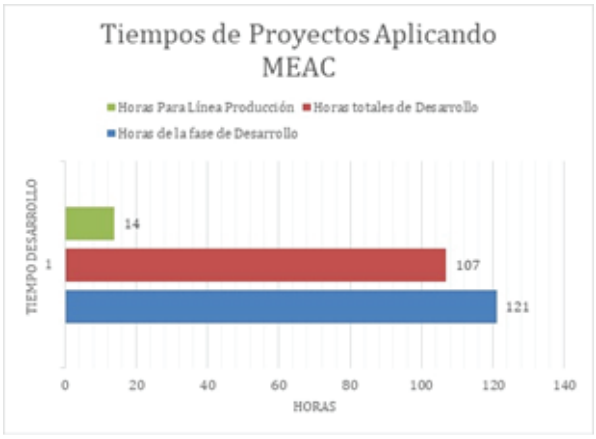

Figure 5: Tiempos del proyecto MEAC.

Como se puede observar en la figura No5, Se establece los tiempos de desarrollo para el proyecto ``Proyecto 2'', en donde se especifica las fases y horas de desarrollo. Con un tiempo establecido de 121 horas en la fase de desarrollo total, siendo el 100% del tiempo invertido, 107 horas de desarrollo que equivalen al 88,43 % de desarrollo, y 14 horas de corte de línea de producción representa al 11,57% del tiempo perdido.

8.2. Proyectos SCRUM vs MEAC SCRUM

A partir del análisis de los tiempos utilizados en el proceso de desarrollo que se han expuesto en el apartado anterior, se pueden establecer comparaciones entre los proyectos aplicados y no aplicados *MEAC*.

8.3. El equipo cumpliendo sus roles

En el proyecto 2 solo existieron dos roles el de Equipo *Scrum* y Product Owner y sus responsabilidades ya se encuentran definidas [15, 24], inicialmente se decidió colocar solo un desarrollador y un gestor para el proyecto, luego se decidió agregar un desarrollador más que cumplió con cierta tareas específicas de desarrollo sin embargo luego salió del proyecto, como se detalló antes el producto Owner se fusionó con el *Scrum* Master de tal manera que su trabajo le permitió al Equipo *Scrum* cumplir específicamente con las tareas de desarrollo.

En el Figura 6, se puede evidenciar que el product owner tiene una participación mucho más activa en el proyecto que esté a su vez va generando actividades a la par

TABLE 4: SCRUM vs MEAC SCRUM.

SCRUM: Proyecto 1	MEAC SCRUM: Proyecto 2	Análisis
hc=$\sum h_{a1+}\ h_{a2+}\ h_{an...}$ **detalle:** hc=horas de corte $\sum h_{a1+}\ h_{a2+}\ h_{an...}$=Sumatoria de las horas actividades extras **aplicando fórmula:** hc=$\sum h_{a1+}\ h_{a2+}\ h_{an...}$ hc=92	hc=$\sum h_{a1+}\ h_{a2+}\ h_{an...}$ **detalle:** hc=horas de corte $\sum h_{a1+}\ h_{a2+}\ h_{an...}$=Sumatoria de las horas actividades extras **aplicando fórmula:** hc=$\sum h_{a1+}\ h_{a2+}\ h_{an...}$ hc=14	Mediante esta fórmula se puede conocer las horas de corte de línea de producción del Proyecto 2 son mínimas respecto a Proyecto 1.
hfd=hc+htd **detalle:***hfd*= horas fase de desarrollo *htd*=horas tiempo desarrollo **aplicando fórmula:** *hfd=92+374* ***hfd=466***	*hfd=hc+htd* **detalle:** *hfd*= horas fase de desarrollo *htd*=horas tiempo desarrollo **aplicando fórmula:** *hfd=14+107* ***hfd=121***	Mediante la fórmula aplicada se puede observar que las horas de fase de desarrollo del Proyecto 2, es mínimo respecto al proyecto 1
htd=hfd-hc detalle: *htd*=horas totales de desarrollo aplicando fórmula: *htd=466-92* ***htd=374***	*htd=hfd-hc* detalle: *htd*=horas totales de desarrollo aplicando fórmula: *htd=121-14* ***htd=107***	Al crear una fórmula para calcular las horas totales desarrollo, se evidenció que el Proyecto 2 minimiza las horas totales de desarrollo, respecto al Proyecto 1, que no aplicar *MEAC*, el tiempo de desarrollo incrementa.

Figure 6: Equipo cumpliendo roles.

del desarrollador lo que le permite continuar con su trabajo sin detener la línea de producción.

8.4. Organizar al Cliente

El *Product Owner* [15]aplicó las actividades definidas para su rol, como establecer el canal de comunicación entre el cliente y el Equipo *Scrum*, siendo el único medio de comunicación entre las partes. De esta manera las inquietudes del cliente fueron resueltas.

9. Conclusiones y Discusión

La experiencia empírica mencionada en secciones anteriores, ha probado que *MEAC* permite minimizar los cortes en la línea de producción de software, ya que toma las

mejores prácticas y roles de *Scrum*, al hacer una analogía se pudo identificar que la metodología *Scrum* se fundamenta en el *control empírico* [5] de procesos de desarrollo, esta es una de las razones que *MEAC* es aplicable a proyectos pequeños de la empresa. Al aplicar *MEAC*, se evidenció que las horas de corte de línea de producción se reducen a un 11.57% versus al proyecto que no se aplicó, donde las horas de corte equivalen al 19.74 %. del tiempo invertido en el desarrollo lo que da como resultado una disminución porcentual del 8.17% del tiempo perdido de desarrollo en actividades distintas a este.

Por lo tanto se puede afirmar que aplicar *MEAC* protege al equipo S*crum* de factores externos e internos que impiden generar de forma continua un producto y ayuda a mejorar los tiempos globales de desarrollo, esto gracias a la redistribución de responsabilidades de los roles de *Scrum*, ya que en proyectos pequeños es muy poco probable que se puedan llenar todos los roles sugeridos por la metodología y esto a su vez es la causante de que el rol de equipo *Scrum* realice actividades que no son propias del mismo y lo desvíen del objetivo principal que es generar el producto. *MEAC* sugirió la forma de hacerlo por medio de la fusión de roles y priorización de tareas dando resultados relevantes en la reducción de cortes de tiempo. Además, se puede reflexionar si estos cortes de tiempo son propios de la metodología *Scrum* en las circunstancias descritas y si estos factores también afectan a otras metodologías.

Podemos ver que al igual que el trabajo presentado por Romano y Delgado [12] el adoptar una metodología a la realidad propia de una empresa genera mejores resultados que aplicarlo de manera Pura, sin embargo es un objetivo futuro de *MEAC* hacerlo lo más genérico posible para ser usado en empresas con realidades similares

La propuesta descrita, es únicamente una experiencia técnica de *Scrum*. Existe un entendimiento bien establecido sobre la aplicación del método ya explicado en la Sección 6. Sin embargo, aún podemos plantearnos preguntas que ayuden a madurar el proceso como: ¿Es posible aplicar este método en empresas de otros tamaños?, ¿qué otras fusiones de roles se pueden hacer? desde el punto vista empírico [25], es factible aplicar esta propuesta en otras empresas con características similares, pues la industria de Software a nivel mundial continúa en crecimiento, buscando una mejor productividad de acuerdo al tamaño de software o tipo de empresa [26], por ellos es importante tener clara la organización interna y externa de los participantes del proyecto y sobre todo siempre tener clara la visión de ``*Siempre Mantener La Línea De Producción Activa*'' de software.

10. Trabajos Futuros

Es interesante conocer y aplicar una metodología para un correcto proceso de desarrollo de software, por ello se plantea seguir probando el Método Empírico de Apoyo Continuo (*MEAC*) en diferentes escenarios y bajo diversos factores como: tamaños de proyectos, números de roles y tipos de metodologías con el fin de conocer cuáles son los resultados de aplicar el fundamento de ``*Siempre Mantener La Línea De Producción Activa*'' en el desarrollo de software, para ello se plantea dos escenarios:

- Madurar *MEAC* en diversos escenarios de la industria: Se planea aplicar *MEAC* en proyectos Reales de la empresa ya no solo limitado a proyectos pequeños, sino también a proyectos medianos y grandes con el fin de conocer si el nivel de eficiencia del equipo *Scrum* en el desarrollo mejora al usar la propuesta o si caso contrato *Scrum* soporta bastante bien el proceso y denota que aplicar *MEAC* es innecesario para este tipo de proyectos.
- Madurar *MEAC* en el campo Académico: Se planea aplicar *MEAC* a proyectos académicos para conocer cuál es su comportamiento en proyectos de estudio, tanto con docentes y estudiantes poniéndolo a prueba sobre escenarios controlados y simulados de tal manera que se pueda predecir su comportamiento en proyectos reales.

El fin último de todo este proceso es madurar *MEAC* generando una metodología estable y comprobada para aplicar en el desarrollo de software de tal manera que se pueda aplicar la misma a escenarios definidos con resultados predecibles generando así productos de software alta calidad, con la menor cantidad pérdida de tiempo posible en el desarrollo.

References

[1] A. Bakhtouchi and R. Rahmouni, ``A Tree Decision Based Approach for Selecting Software Development Methodology,'' in *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, El Oued, 2018, pp. 211--216.

[2] N. Keshta and Y. Morgan, ``Comparison between traditional plan-based and agile software processes according to team size & project domain (A systematic literature review),'' in *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, 2017, pp. 567--575.

[3] R. Bin-Hezam, A. Bin-Essa, and N. F. Abubacker, ``Is the Agile Development Method the Way to Go for Small to Medium Enterprises (SMEs) In Saudi Arabia?,'' in *2018 21st Saudi Computer Society National Computer Conference (NCC)*, Riyadh, 2018, pp. 1--6.

[4] B. M. Montero, H. V. Cevallos, and J. D. Cuesta, ``Agile methodologies against traditional methods in the software development process.,'' p. 10.

[5] J. P. Z. Gamboa and C. A. L. Arreaga, ``Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software. Evolution of the Methodologies and Models used in Software Development.,'' vol. 3, no. 10, p. 14, 2018.

[6] L. R. Vijayasarathy and C. W. Butler, ``Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter?,'' *IEEE Softw.*, vol. 33, no. 5, pp. 86--94, Sep. 2016.

[7] K. Curcio, R. Santana, S. Reinehr, and A. Malucelli, ``Usability in agile software development: A tertiary study,'' *Comput. Stand. Interfaces*, vol. 64, pp. 61--77, May 2019.

[8] E. T. Mushashu and J. S. Mtebe, ``Investigating Software Development Methodologies and Practices in Software Industry in Tanzania,'' in *2019 IST-Africa Week Conference (IST-Africa)*, Nairobi, Kenya, 2019, pp. 1--11.

[9] H. Alahyari, R. Berntsson Svensson, and T. Gorschek, ``A study of value in agile software development organizations,'' *J. Syst. Softw.*, vol. 125, pp. 271--288, Mar. 2017.

[10] L. N. M. Velandia and W. M. L. López, ``ESCOGER UNA METODOLOGÍA PARA DESARROLLAR SOFTWARE, DIFÍCIL DECISIÓN,'' *Rev. Educ. En Ing.*, vol. 10, p. 12, 2015.

[11] K. Kaur and A. Jajoo, ``Applying Agile Methodologies in Industry Projects: Benefits and Challenges,'' in *2015 International Conference on Computing Communication Control and Automation*, Pune, India, 2015, pp. 832--836.

[12] B. L. Romano and A. D. da Silva, ``Project Management Using the Scrum Agile Method: A Case Study within a Small Enterprise,'' in *2015 12th International Conference on Information Technology - New Generations*, Las Vegas, NV, USA, 2015, pp. 774--776.

[13] A. Srivastava, S. Bhardwaj, and S. Saraswat, ``SCRUM model for agile methodology,'' in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, Greater Noida, 2017, pp. 864--869.

[14] J. D. Y. González and S. G. Gómez, ``Revisión sistemática acerca de la implementación de metodologías ágiles y otros modelos en micro, pequeñas y medianas empresas de software,'' p. 16.

[15] T. Miles-Board, ``Poster: An Empirical Study of the Product Owner Role in Scrum,'' p. 2.

[16] A. M. G. Rodríguez, Y. T. Casañola, and A. P. Vergara, ``Optimización de estados en la mejora de procesos de software,'' p. 20.

[17] ``Establecimiento del estado del arte sobre el aligeramiento de procesos de software,'' *RISTI - Rev. Ibérica Sist. E Tecnol. Informação*, no. 17, Mar. 2016.

[18] A. N. Cadavid, ``Revisión de metodologías ágiles para el desarrollo de software,'' *Prospectiva*, vol. 11, no. 2, p. 30, Sep. 2013.

[19] G. P. Tomaselli, C. J. Acuña, M. Estayno, and C. Lenkovich, ``SCRUM: Una revisión de la literatura,'' p. 11.

[20] C. Rodríguez and R. Dorado, ``¿Por qué implementar Scrum?,'' *Rev. Ontare*, vol. 3, no. 1, p. 125, Oct. 2015.

[21] G. Matturro, F. Cordovés, and M. Solari, ``An exploratory study of the role of Product Owner in industrial practice,'' p. 12.

[22] V. P. Díaz-Narváez V.P. and A. Calzadilla-Núñez A., ``Artículos científicos, tipos de investigación y productividad científica en las Ciencias de la Salud,'' *Cienc. Salud*, vol. 14, no. 1, pp. 115--121, Feb. 2016.

[23] M. P. Polo, ``Ceremonial y protocolo: métodos y técnicas de investigación científica,'' p. 21, 2015.

[24] J. Vlietland and H. van Vliet, ``Towards a governance framework for chains of Scrum teams,'' *Inf. Softw. Technol.*, vol. 57, pp. 52--65, Jan. 2015.

[25] D. Méndez Fernández and J.-H. Passoth, ``Empirical software engineering: From discipline to interdiscipline,'' *J. Syst. Softw.*, vol. 148, pp. 170--179, Feb. 2019.

[26] ``Industria de Software,'' p. 45.