UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

"MODELO DE GESTIÓN DE DESARROLLO DE SOFTWARE ÁGIL MEDIANTE SCRUM Y KANBAN SOBRE LA PROGRAMACIÓN EXTREMA, 2019"

Tesis presentada por: Eder Solórzano Huallanca.

Para optar el título profesional de: Ingeniero de Sistemas

Tipo de investigación: Observacional, Retrospectivo y Transversal

Área de investigación: Ingeniería de Software

Asesor: Mg. Ing. Hubner Janampa Patilla.

Ayacucho - Perú

2020

DEDICATORIA

Primeramente a Dios por guiarme cada día, darme salud y acompañarme en este camino de vida.

A mi madre por haberme criado de la mejor forma, por sus consejos, su paciencia, sus exigencias y su apoyo incondicional durante toda mi vida; a mi hermana por su constante apoyo y preocupación, a Jazmín por su apoyo y deseo perseverante en la realización durante todo el desarrollo de este proyecto.

AGRADECIMIENTO

A mi alma mater, Universidad Nacional de San Cristóbal de Huamanga, a cada uno de los docentes de la Escuela Profesional de Ingeniería de Sistemas, por su constante labor en la docencia universitaria que ayudaron mucho en mi vida universitaria, a mi mentor Mg. Ing. Hubner Janampa Patilla, por su apoyo, asesoramiento y guía para la realización de esta tesina.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

No se encuentran elementos de tabla de ilustraciones.

RESUMEN

Las metodologías agiles nos sirven para elaborar un software, por eso siempre es necesario elegir una metodología. En el contexto de los proyectos de desarrollo, los requisitos son cambiantes, estas metodologías, nos facilitan aminorar los tiempos de progreso sin tener que descuidar la elevada calidad del producto, teniendo como base las metodologías usaremos en este trabajo, Programación Extrema (XP), Scrum y Kanban. Xp dirigido al área netamente de programación, Scrum, al ordenamiento y administración y, Kanban, gestiona el óptimo flujo de cada trabajo dentro de un proceso, por tanto estas metodologías pueden añadirse y alcanzar un resultado de destacada cualidad.

El tipo de investigación es observacional, retrospectivo y transversal. Teniendo como máximo alcance la implementación de un modelo de desarrollo de software utilizando las tres metodologías (Programación Extrema sobre Scrum y Kanban) para poder permitir de mejor manera la gestión de software ágil.

En la actual investigación se busca abstraer los conceptos relevantes de la Programación Extrema, Scrum y Kanban, para mostrar un prototipo híbrido que proporcionará desarrollar un software ágil.

Palabras claves: Programación Extrema en Scrum y Kanban, Gestión de proyecto ágil, Pruebas Unitarias Continuas, metodologías ágiles, Modelo Híbrido.

INTRODUCCIÓN

En el contexto de actualidad en desarrollo de software, estas metodologías están siendo cada vez más utilizadas, por la cual resulta hasta obligatorio el elegir una metodología como mínimo, pero, el problema surge ya en la elaboración del proyecto, si se pone énfasis en el desarrollo del software en si o en las prácticas de programación. Scrum se relaciona con la primera, administrando el proceso de desarrollo, mientras tanto XP se relaciona con lo segundo; y Kanban nos va a gestionar el óptimo flujo de trabajo en cada proceso del proyecto de desarrollo.

Las compañías e instituciones de desarrollo de software, buscan que un producto tenga calidad sin descuidar los fundamentos de gestión y prácticas de organización. Por la cual como objetivo y razón de ser de esta tesis, es plantear y formular una plantilla de desarrollo de software de la Programación Extrema en Scrum y Kanban.

Esta metodología híbrida se deriva a partir del análisis exhaustivo de las metodologías ágiles: Programación Extrema, Scrum y Kanban. Estas metodologías son ampliamente utilizadas en el desarrollo de aplicaciones en la actualidad.

Los principales objetivos son: a) Proponer el modelo de gestión de desarrollo de software ágil mediante Scrum y Kanban para que se adapte al modelo iterativo e incremental de la Programación Extrema. b) Proponer el modelo de gestión de desarrollo de software ágil mediante Scrum y Kanban para que se adapte a las pruebas unitarias de la Programación Extrema. c) Proponer el modelo de gestión de desarrollo de software ágil mediante Scrum y Kanban para que se adapte a la re factorización de código de la Programación Extrema. d) Proponer el modelo de gestión de desarrollo de software ágil mediante Scrum y Kanban para que se adapte a la corrección de errores y código compartido de la Programación Extrema.

CAPÍTULO I

PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1. DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

Para Sommerville (2011), Las empresas que desarrollan software siempre utilizan metodologías ágiles como el Iconix, Scrum, Programación Extrema, Kanban, etc., preocupándose desde las primeras etapas de desarrollo hasta después de haber terminado el producto con el mantenimiento que se le da a las aplicaciones, pero no solamente se debe de prestar atención a la elaboración del producto como tal, sino que también a la administración de los procesos, creación de herramientas, aplicando teorías y procedimientos con el fin de obtener calidad y cumplir con las fechas establecidas de entrega. Teniendo en cuenta que el cliente siempre querrá un buen producto en un tiempo apresurado, por eso es necesidad la de utilizar y combinar metodologías ágiles para el desarrollo de un producto. Pero, cada metodología que queramos usar tiene distintas fases y procedimientos como indica Kniberg (2007) que, Scrum se orienta en las prácticas de ordenamiento y gestión, entretanto que XP se concentra más en las prácticas de programación, Klipp (2014), afirma que Kanban proporciona gestión y óptimo flujo de trabajo dentro del proceso. Hubieron autores que tenían la propuesta de utilizar un enfoque hibrido, ya que como se dijo cada metodología tiene sus propias formas de conseguir el producto final asegurando el tiempo y la inversión. Como es el caso de DeMarco y Boehm (2002), que decían que la metodología híbrida elegida debería de contener ciertas técnicas que mantuviesen la organización a largo plazo.

La gestión de proyectos de software es una parte esencial de la ingeniería de software. Los proyectos necesitan administrarse porque la ingeniería de software profesional está sujeta siempre a restricciones organizacionales de presupuesto y fecha. El equipo de trabajo tiene la función y obligación de velar por la calidad del producto, así como de poder saber superar las restricciones que todo proyecto de desarrollo supone. El saber cómo gestionar un proyecto de desarrollo y ponerlo en práctica no garantiza el éxito, pero una mala gestión sí que trae resultados

negativos para un proyecto como baja calidad de código, no cumplir con los compromisos y tiempos, no satisfacer al cliente. (Sommerville, 2011).

En la elaboración de un producto, tenemos metodologías ágiles como Xp que está enfocada al desarrollo del software en sí, o tenemos otras enfocadas a la gestión del proyecto que ayudará en la buena organización de los procesos como es el caso de Scrum y Kanban que gestiona un óptimo flujo de trabajo dentro del proceso; en consecuencia, estás metodologías pueden combinarse y buscar el complemento de sus etapas y procesos, integrándose de tal manera que se encuentre la armonía, con el fin de obtener un producto de calidad, elaborado con los mejores procedimientos.

1.2. FORMULACIÓN DEL PROBLEMA DE INVESTIGACIÓN

1.2.1. PROBLEMA PRINCIPAL

¿Cómo el modelo de gestión de desarrollo de **software ágil mediante Scrum y Kanban se adapta a la Programación Extrema**, 2019?

1.2.2. PROBLEMAS SECUNDARIOS

a. ¿Cómo el modelo de gestión de desarrollo de **software ágil mediante Scrum y Kanban se adapta al modelo iterativo e incremental de la Programación Extrema**, 2019?
b. ¿Cómo el modelo de gestión de desarrollo de **software ágil mediante Scrum y Kanban se adapta a las pruebas unitarias de la Programación Extrema**, 2019?
c. ¿Cómo el modelo de gestión de desarrollo de **software ágil mediante Scrum y Kanban se adapta a la re factorización de código de la Programación Extrema**, 2019?
d. ¿Cómo el modelo de gestión de desarrollo de **software ágil mediante Scrum y Kanban se adapta a la corrección de errores y código compartido de la Programación Extrema**, 2019?

1.3. OBJETIVOS DE LA INVESTIGACIÓN

1.3.1. OBJETIVO GENERAL

Implementar el modelo de gestión de desarrollo de **software ágil mediante Scrum y Kanban a la Programación Extrema**, 2019.

1.3.2. OBJETIVOS ESPECÍFICOS

a. Proponer el modelo de gestión de desarrollo de **software ágil mediante Scrum y Kanban para que se adapte al modelo iterativo e incremental de la Programación Extrema**, 2019.

b. Proponer el modelo de gestión de desarrollo de **software ágil mediante Scrum y Kanban para que se adapte a las pruebas unitarias de la Programación Extrema**, 2019.

c. Proponer el modelo de gestión de desarrollo de **software ágil mediante Scrum y Kanban para que se adapte a la re factorización de código de la Programación Extrema**, 2019.

d. Proponer el modelo de gestión de desarrollo de **software ágil mediante Scrum y Kanban para que se adapte a la corrección de errores y código compartido de la Programación Extrema**, 2019.

1.4. JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

1.4.1. IMPORTANCIA DEL TEMA

IMPORTANCIA TÉCNICA

Esta propuesta de metodología híbrida busca contribuir a que los desarrolladores utilicen este modelo como una metodología para desarrollar software ágil de calidad, para lo cual se ha adoptado las prácticas, teorías y procedimientos más sobresalientes de programación de la metodología XP y las prácticas de gestión y administración de las metodologías Scrum y Kanban.

"Los métodos híbridos basan su existencia en las debilidades de los métodos anteriormente nombrados, con la finalidad de crear un método robusto pero al mismo tiempo flexible, que combine las bondades de dos o más metodologías ágiles." (Leiva & Villalobos, 2015).

IMPORTANCIA ECONÓMICA

La adecuada gestión de software ágil, permitirá estimar el tamaño de lo que estamos construyendo y medir la velocidad en el que podemos hacer el trabajo. Con esta información podemos estimar el costo correspondiente, asimismo el modelo de desarrollo de software usando Scrum y Kanban sobre la Programación Extrema, reducirá el costo de los cambios generados en etapas posteriores en el desarrollo del proyecto.

Para (Pressman, 2010), la "ingeniería de software es el establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales."

1.4.2. JUSTIFICACIÓN

Para Vlaanderen, Jansen, Brinkkemper y Jasper (2011), observan que en los últimos años, el trabajo de desarrollar software tiene un entorno cambiante, ya sea de requisitos, requerimientos, prácticas, por lo cual se permite integrar al grupo de

trabajo a los clientes, que son ellos que usarán el resultado del desarrollo y como punto importante elegir al producto por encima de la documentación propia de cada metodología, todo estos puntos son inherentes a los principios ágiles, representando en el marco de la metodología de desarrollo de software como innovaciones en estos últimos años, siendo obligatorio tener dos ideas importantes en lo que a metodología ágil se refiere: Prácticas de programación y la parte de gestión de software ágil.

Kniberg (2007), afirma que Scrum se enfoca en las prácticas de organización y gestión, mientras que la programación extrema se centra más en las prácticas de programación, Klipp (2014), afirma que Kanban proporciona gestión y óptimo flujo de trabajo dentro del proceso. Este trabajo de investigación, busca plantear un modelo de metodología híbrida para el desarrollo de software, integrando los procesos de Xp, Scrum y Kanban, desde una perspectiva de reducción de tiempo y costo, con una adecuada planificación durante el desarrollo de la misma.

1.4.3. DELIMITACIÓN

Se está planteando un modelo de metodología híbrida con el uso netamente de los principios, teorías y prácticas de las metodologías Xp, integrándolas con Scrum y Kanban que permitirá un desarrollo de calidad con gestión y organización optimizando el flujo de trabajo dentro del proceso.

CAPÍTULO II

REVISIÓN DE LA LITERATURA

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Los autores Carrasco, Ocampo, Ulloa y Azcona (2019), en su trabajo de investigación denominada "Metodología híbrida de desarrollo de software combinando XP y Scrum", de la Universidad Regional Autónoma de Los Andes de Ecuador, concluyen que la combinación de la metodología XP y el marco de trabajo Scrum son de gran ayuda en el desarrollo de software. Los procesos y el ciclo de vida ideal que propone XP junto con los eventos y artefactos Scrum aportaron vital importancia para que el desarrollo sea versátil y limpio, de manera que permita realizar cualquier cambio en el producto que surja a lo largo de los Sprints, sin necesidad de comprometer las funcionalidades ya desarrolladas.

Para el autor Gamboa (2014), en su trabajo de investigación denominada "Aumento de la productividad en la gestión de proyectos, utilizando una metodología ágil aplicada en una fábrica de software en la ciudad de Guayaquil", de la Universidad Espíritu Santos de Ecuador, afirma que uno de los principales problemas que enfrentan actualmente las empresas que se dedican al desarrollo de software, es la lucha constante contra el tiempo. El cliente siempre buscará calidad en el menor tiempo posible y para la disminución de tiempos se requiere de más personal capacitado. Por eso y más factores hoy las empresas que desarrollan software se ven en la necesidad de implementar métodos que ayuden en la gestión de sus proyectos.

2.2. MARCO TEÓRICO

2.2.1. GESTIÓN DE SOFTWARE ÁGIL

La ingeniería de software comprende múltiples actividades que incluyen desde la captura de requisitos, análisis, gestión de procesos, diseño, codificación y pruebas. Hay tareas que no se pueden terminar haciéndolas de manera individual sino que implican la colaboración de cierto número de personas para terminar de la mejor manera, de forma eficiente dicha actividad, entonces la Administración

de software o Administración de proyectos de software, se entiende por la coordinación de este grupo de personas con el fin de poder administrar las actividades en un proyecto software. La gestión de proyectos comprende cinco grandes bloques: planificar, gestionar personal, organizar, controlar y dirigir (Tuya, Ramos y Dolado, 2007).

Sommerville (2010) afirma que la "gestión de proyectos de software es una parte esencial de la ingeniería de software. Esta no garantiza el éxito del proyecto, sin embargo, usualmente una mala gestión lleva al fracaso". Una mala gestión, resulta en la falta de cumplimiento de los tiempos establecidos con el cliente además de no cumplir con las expectativas de este, generalmente el éxito de un proyecto de desarrollo depende de la gestión del mismo, teniendo como límites los ya siempre presentes restricciones de tiempo y presupuesto.

La gestión de proyectos ágiles como tal, implican un extenso número de principios, de los cuales tenemos los más importantes como pilares de gestión: adaptarse al cambio, cumplir con las especificaciones del cliente otorgando valor para él y cada cierto tiempo previamente pactado brindar un módulo funcional del software de manera concurrente. (Fitsilis, 2008)

Como un resumen de los principios del manifiesto ágil, mencionados en (Agilemanifesto, 2020), dicen: "estamos descubriendo mejores formas de desarrollar software haciéndolo y ayudando a otros a hacerlo. A través de este trabajo hemos llegado a valorar:

- **Individuos e interacciones** sobre procesos y herramientas.
- **Software de trabajo** sobre documentación completa.
- **Colaboración del cliente** sobre negociación de contrato.
- **Responder al cambio** sobre seguir un plan.

Es decir, si bien hay valor en los elementos de la derecha, valoramos más los elementos de la izquierda."

2.2.2. MÉTODOS Y TÉCNICAS

2.2.2.1. PROGRAMACIÓN EXTREMA (XP)

Pardo (2010), nos dice: Siendo sino el más usado de las metodologías ágiles, Xp se sitúa como uno de los más populares y utilizados por desarrolladores e instituciones de desarrollo, cuya característica principal reside en adaptarse que en ser previsible. Aquellos desarrolladores que lo utilizan afirman que los requisitos cambiantes son naturales, inevitables y hasta deseables en un proyecto de desarrollo, entonces el poder adaptarse a estos cambios en cualquier parte del mismo, es por la cual se elige y trabaja con Xp, que teniendo un modelo único y sin cambios en los requisitos con el cual trabajar desde el inicio, perdiendo tiempo y recursos en solucionar los cambios que se den en el transcurso del desarrollo.

Según Canos (2004), es "una disciplina de desarrollo de software basada en los valores de simplicidad, comunicación, retroalimentación y valor. En XP cada participante del proyecto es una parte integral del equipo."
El equipo se junta con el cliente quien trabajará y estará disponible para ellos, todo el tiempo que dure el desarrollo, a través del acuerdo de tiempos, entre el cliente y el equipo, Xp brinda una forma predictiva de terminar un proyecto para un tiempo previsto, con un producto de valor que fue alcanzado gracias a los entregables continuos y constantes, que pasaron las pruebas de testeo, ayudados con la re factorización de código, el trabajo en pares y el diseño simple que permite Xp, viendo siempre satisfacer las expectativas del cliente, gracias a los requerimientos brindados por él.

Sommerville (2011), sustenta, que como parte del enfoque de Xp, promueve una elaboración iterativa del proyecto, conjuntamente con la participación activa del cliente, utilizando en el desarrollo del producto las buenas prácticas, como parte de ser un método ágil de desarrollo.

A. VALORES DE LA PROGRAMACIÓN EXTREMA

Según Beck y Andrés (2005), nos comenta que Xp es una doctrina para el desarrollo de software que tiene como valores la comunicación, un feedback, simplicidad, valor para cambiar o desechar código y respeto.

a. COMUNICACIÓN

Como valor de Xp, promueve la fluidez en la comunicación de cada uno de los miembros del equipo de desarrollo, ubicándolos en un mismo ambiente, con el fin de notar los gestos del cliente, las expresiones, estados anímicos, humor, cansancio de cada uno de los integrantes del equipo. (Fernández, 2014).

"Muchos de los problemas que existen en proyectos de software, se deben a problemas de comunicación entre las personas. La comunicación permanente es fundamental en XP, dado que los artefactos son pocos." Como parte de estar en un ambiente todos los actores del proyecto de software, se mejora la comunicación directa, para consultar dudas, ampliar detalles de las historias de usuario, etc. (Beck, 1999)

b. SIMPLICIDAD

Xp promueve durante cada etapa de desarrollo, la sencillez y simplicidad en diseño, código, procesos, prácticas y demás. Gracias a este valor, cada uno de los integrantes del equipo podrá entender y comprender las líneas de código del producto para poder implementar mejoras continuas en todo el transcurso del proyecto. (Beck, 1999).

Este valor promueve que los cambios que se harán al proyecto sean en costos, los más bajos posibles, por eso pide implementar código sencillo y entendible. (Canos, 2004).

c. REALIMENTACIÓN

Xp promueve una retroalimentación constante, si algún integrante del equipo encuentra algún requerimiento inadecuado, este pasa a ser revisado nuevamente. (Fernández, 2014).

Con la participación del cliente en todas las etapas, éste brinda sus impresiones de las historias de usuario hechas, con el fin de mejorar en la siguiente iteración,

cumpliendo así sus necesidades del producto. Existen las pruebas unitarias que como retroalimentación constante y permanente contribuyen a la calidad del producto. (Beck, 1999).

d. CORAJE

Con Xp se rompe el paradigma de que si una línea de código funciona ya no debería de tocarse, pero esta metodología no se detiene ahí sino que busca mejorar lo que ya está bien y esto para todo programador, es una decisión difícil de tomar (Fernández, 2014).

En cada proyecto de desarrollo se maneja un presupuesto y un tiempo establecido, Xp al tener mejora constante, no escatima en rehacer código sea cual sea la fase del ciclo que se encuentre el proyecto, así se modificar una línea o todo el código. (Beck, 1999).

e. RESPETO

Entre cada uno de los miembros del equipo prima el respeto, así sólo se contribuya con entusiasmo, respeto con el cliente y de el para el equipo, respeto también desde el nivel jerárquico más alto para el equipo brindándole autonomía en su trabajo. Respeto entre quien sabe y conoce más con quién sepa menos. Es un valor importante que debe de instaurarse en todo aspecto y más aún cuando se trabaje con un grupo de personas.(Wells, 2019).

"Es el valor que resume los otros cuatro y sin el que los demás no funcionarán. Si no hay respeto entre los miembros del equipo y éstos no respetan al proyecto y a sus compañeros, difícilmente se obtendrá un feedback correcto." Xp no maneja jerarquías de posición, Xp promueve el trabajo de equipo, nadie es más que otra persona, todos aportan al desarrollo, ni el gerente es más importante que el programador ni este con el ingeniero de software, con el fin de tomar rápidas y mejores decisiones. (Martel, 2018).

B. PRINCIPIOS DE LA PROGRAMACIÓN EXTREMA

Somerville (2005) brinda las siguientes prácticas y principios en Xp:

a. Al conversar con el cliente se tiene una lista de requerimientos que son transformadas en historias de usuario, éstas al ser elaboradas deben de indicar el tiempo en el cual va a terminarse y la prioridad que tiene para la iteración, después estas actividades serán divididas en tareas de desarrollo.

b. Desde el primer momento se promueve las entregas incrementales y pequeñas, se comienza a implementar funcionalidades mínimas que den valor al producto. Son frecuentes, constantes y se añaden gradualmente.

c. Sólo se hace lo necesario, un diseño sencillo para cumplir con lo establecido, no más.

d. Xp brinda un conjunto de pruebas por las que cada mejora debe de pasar, generalmente ya se tiene automatizada estas pruebas, y antes de lanzar cada mejora se debe de tener el visto bueno de las pruebas sometidas.

e. Re factorización, es obligación y responsabilidad de cada miembro del equipo en especial los desarrolladores el de re factorizar el código, si en el día a día se encontrase errores o mejoras de código, Xp nos permite hacer con el fin de tener un código con sencillez y siempre mejorado.

f. Xp promueve el desarrollo en pareja, con el fin de mejorar las líneas de código, ya que uno será el veedor del otro desarrollador, que conllevará a la disminución de errores en código e implementar más rápido las funcionalidades al implementar sólo lo necesario.

g. Al promover Xp el trabajo en parejas, cada una de éstas, interviene en el desarrollo durante todo el proyecto, entonces se rompe el mito de que si este código yo lo hice entonces me pertenece, por tanto cada miembro del equipo puede modificar código porque todos poseen el código, no hay secretos, no hay propiedades de código.

h. A través de las pruebas unitarias, Xp realiza un control de código y funcionalidad, entonces este se integra a todo el sistema en ejecución de una manera continua y constante con el fin tener los incrementos listo y sin errores.

i. Tener un rimo constante de trabajo durante la realización de todo el proyecto, sin tener un incremento saturado de horas extras, que al final traen consecuencias para el proyecto con respecto a la calidad del código ya que se verá mermado disminuyendo la productividad.

j. Es preferible y hasta necesario, que el cliente o un representante del mismo este siempre presente en la realización de todo el proyecto, que sea capaz de tomar decisiones, convirtiéndose así en un integrante más del equipo de trabajo y ayudará en la solución de dudas respecto a los requerimientos.

C. ROLES DE LA PROGRAMACIÓN EXTREMA

Según Beck (2004), los roles de XP son:

Programador. Persona responsable de la elaboración del código y quien redacta las pruebas unitarias que deberá pasar cada funcionalidad realizada. Debiendo existir siempre comunicación fluida entre los integrantes del equipo y los programadores.

Cliente. Comienza con describir los requerimientos y elaboración de historias de usuario, manifestando cuál historia será implementada antes que otra.

Encargado de pruebas (tester). Persona que va a poner a prueba las funcionalidades hechas por parte del equipo de desarrollo, y verificar que cada prueba unitaria implementada funcione correctamente. Después de las pruebas tendrá la obligación de informar el resultado de las mismas al equipo de trabajo.

Encargado de seguimiento (traker). Persona que tiene la responsabilidad de hacer seguimiento a cada una de las funcionalidades, tareas e iteraciones hechas, tendrá también la obligación de retroalimentar al equipo y tener un registro de las pruebas realizadas.

Entrenador (coach). Persona que brinda orientación y está en constante relación con los integrantes del equipo, revisando que se utilicen las correctas prácticas de Xp, siendo responsable de la ejecución de todo el progreso del proyecto.

Consultor. En el transcurrir de la realización del proyecto, surgirán temas que desconocen los miembros del equipo y el implementarlas resultará riesgoso para el proyecto, entonces se requiere a un miembro que no pertenezca al equipo con el fin de consultar y tener los conocimientos exactos.

Gestor (BigBoss). Persona que tiene la función de ser el intermediario entre el equipo de desarrollo y el cliente, cumplirá la función de coordinación.

D. FASES DEL PROCESO ÁGIL XP

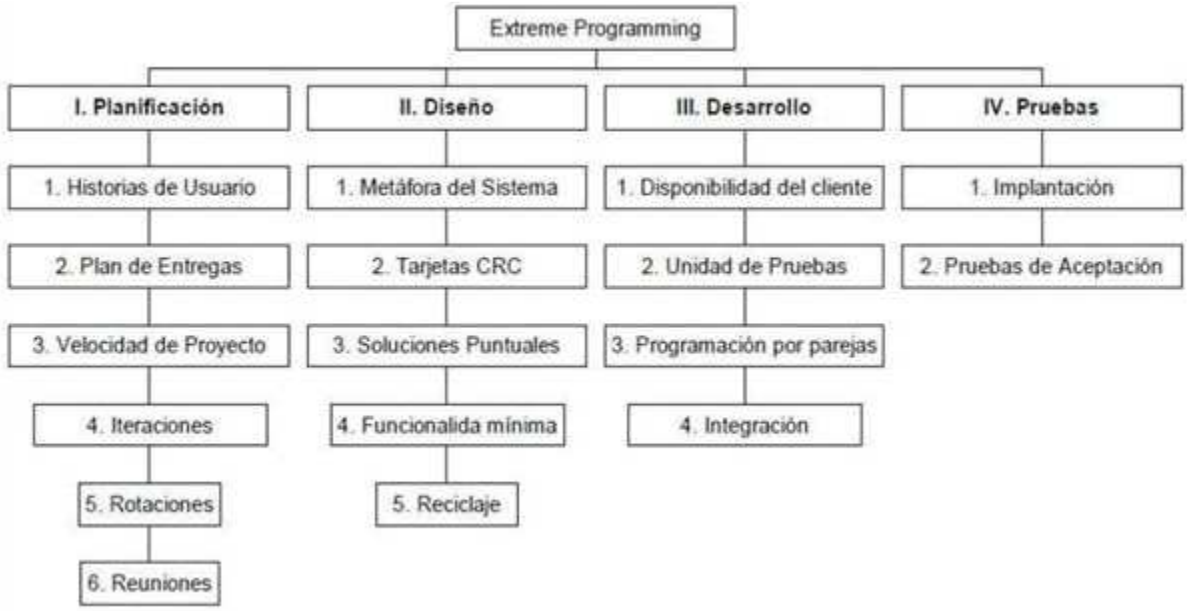


Figura 1. Fases de la metodología XP (Jeffries, Anderson y Hendrickson, 2001)

A. PLANIFICACIÓN

Según Jeffries, Anderson y Hendrickson (2001), el valor de la comunicación plantea que sea fluida entre todos los miembros del equipo, siendo estas definiciones las siguientes:

a. Historia de usuario: En Iconix usamos los casos de uso, en Xp, las historias de usuario, que al ser los requerimientos del cliente deben de estar escritas en el lenguaje que ellos usan, sin términos técnicos, que son básicamente lo que el sistema quiere que haga y tenga para ellos, se recomienda que sean detalladas con el fin de ayudar a los otros miembros del equipo en su estimación, como parte de que el cliente siempre será parte del proyecto entonces los integrantes consultarán sus dudas directamente con él. (Jeffries, Anderson y Hendrickson, 2001).

b. Plan de entregas (Release Plan): Como parte de la comunicación fluida entre todos los miembros del equipo, se establece un cronograma que indicará las fechas de entrega de las historias que han sido agrupadas en la reunión del Planning game o juego de planeamiento, será determinada por el cliente quien indicará las historias que deben de agruparse y las funcionalidades primeras en implementarse. (Beck, 1999).

c. Velocidad del proyecto: Nos servirá de guía para todas las iteraciones del grupo, estas iteraciones están dadas por la cantidad de historias de usuario que contiene cada una, que será nuestro ritmo sostenible durante todo el proyecto sin incrementar demás o quitar historias, determinará la rapidez de desarrollo del proyecto, es necesario evaluar el número de historias en una iteración después de 3 o 4 iteraciones para comprobar la velocidad del cumplimiento de cada iteración. Si sucede que no es la adecuada para el cliente se debe de tener una reunión conocida como el Release Plan.

d. Plan de iteraciones: Como ya tenemos un cronograma de iteración ahora toca implementar y desarrollar una historia de usuario, respetando como esta en el cronograma. En cada iteración se llevará a cabo una reunión para esta actividad. (Beck, 1999).

e. Rotaciones: Como se dijo líneas arriba el trabajar en pares hará que todas los integrantes conozcan todo el proyecto, así evitaremos que haya integrantes que desconozcan lo que se está haciendo.

f. Reuniones: Es necesario que los desarrolladores se reúnan diariamente y expongan sus problemas, soluciones e ideas de forma conjunta. Las reuniones tienen que ser fluidas y todo el mundo tiene que tener voz y voto.

B. DISEÑO

Según Jefferies, Anderson y Hendrickson (2001), como parte de los valores de Xp tenemos que las tareas y actividades deben ser lo más simples posibles. Tenemos las siguientes definiciones:

a. Simplicidad: Gracias a lo simple y claro que sean los procesos se podrá implementar más rápidamente lo cual significará menos esfuerzo, menos tiempo y menos costo para el proyecto. No debemos de complicar o confundir a los otros miembros del grupo en la realización de tareas, actividades y procesos.

b. Metáfora del sistema: En la realización de todo proyecto surgen temas, conceptos y términos nuevos que algunos de los integrantes desconocen. El no complicar con nombres los métodos, clases y el código en sí ayudará para que cada uno de los integrantes comprenda y pueda re utilizar el código.

c. Tarjetas CRC: (Clase, Responsabilidad y Colaboración) permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica. Para implementar las funcionalidades, se encuentran por parte del programador, organizados en clases y objetos. Teniendo una plantilla en donde se aprecia que el título será el nombre de clase (parte superior), en la columna izquierda notaremos que se encuentran las responsabilidades, los objetivos que debe de cumplir y en la parte derecha con qué otras clases se relaciona.

d. Soluciones puntuales: Existen programas de prueba “spike” que ayudarán a corregir problemas y evaluar soluciones.

e. Funcionalidad mínima: En el transcurso del desarrollo del sistema se están aplicando técnicas y prácticas con el fin de aminorar tiempos, costos y esfuerzo, por lo cual sólo se debe de implementar lo que se necesita, sin perder hombres-hora en la realización de otras cosas que no se usarán para dicha funcionalidad.

f. Re factorizar: Al tener que re utilizar código ya desarrollado puede contar con línea no usadas que generan a lo largo inestabilidad y malos diseños, entonces es buena práctica re factorizar el código sin cambiar para el objetivo por el cual fue implementado, pudiéndose cambiar la estructura del código.

C. DESARROLLO

Según Jeffries, Anderson y Hendrickson (2001), contamos con las definiciones respectivas:

a. Disponibilidad del cliente: Al tener a todos los integrantes en un mismo ambiente y siendo el cliente uno más del equipo, se lo tiene a disposición para las consultas y así obtener un producto acorde a sus necesidades.

b. Unidad de prueba: Como parte de las prácticas que Xp tiene, están las diferentes pruebas unitarias que se le hace a cada funcionalidad antes de implementarla, y aún también al integrarla al sistema se realizan pruebas, con el fin de obtener un producto de calidad, gracias al valor de la simplicidad se debe de realizar código simple para poder pasar estar pruebas.

c. Programación por parejas: La contratación de nuevo personal para un proyecto supone ciertamente el uso de recursos financieros y en ciertos casos, tiempos. Gracias a esta programación por pares se comenten menos errores, por lo tanto habrá menos cosas por corregir, menos tiempo invertido en estas correcciones que supondrán menos costos para el proyectos, teniendo así equilibrio de gastos y tiempo.

d. Integración: Al terminar una funcionalidad y pasar esta por las pruebas debidas, tenemos un código limpio y libre de errores, entonces se debe de integrar al todo, que será conocida como nuestra última versión, por lo que no se debe de hacer cambios en versiones obsoletas de las funcionalidades que supondría el gasto de tiempo, dinero y esfuerzo en lo que seguramente traería problemas para el proyecto.

D. PRUEBAS

Los autores Jeffries, Anderson y Hendrickson (2001), proponen las siguientes definiciones:

A. Pruebas unitarias: Cada funcionalidad antes de ser implementa tiene que pasar por las respectivas pruebas unitarias para tener un código sin errores, estas pruebas ya deben de ser desarrolladas antes para tenerlas listas.

B. Detección y corrección de errores: A consecuencia de las pruebas unitarias y la integración, se pueden encontrar errores en los códigos y funcionalidades, es por eso que es importante el detectar estos antes de una integración, mejorando así la calidad del sistema, aminorando costos, tiempos y esfuerzos, es una actividad constante y permanente en la realización de todo proyecto de desarrollo de software.

C. Implantación: La implementación e integración del código y las funcionalidades, siempre deben de ser a consecuencia del uso de las pruebas unitarias. No se puede implantar código con errores porque traerán graves consecuencias para el proyecto.

D. Pruebas de aceptación: Al tener ya una funcionalidad lista producto de la realización de una historia de usuario, también debería de ser sometida a pruebas de aceptación, que vienen a ser pruebas o situaciones que el mismo cliente propone para que esta historia de usuario funcione correctamente, siendo responsabilidad del cliente el aceptarlas o no.

E. ARTEFACTOS DE XP

HISTORIAS DE USUARIO

"Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales." Xp es una metodología ágil, dinámica en donde las historias de usuarios no son guías obligatorias a seguir, sino que siendo parte de metodología ágil pueden cambiarse, eliminarse o ser reemplazadas por otras, estas historias hechas en un lenguaje del cliente, son descompuestas en tareas para que los programadores implementen la funcionalidad. (Letelier y Penadés, 2006).

HISTORIA DE USUARIO	
Numero:	**Usuario:**
Nombre Historia:	
Prioridad en Negocio:	**Riesgo en Desarrollo:**
Puntos Estimados:	**Iteración Asignada:**
Programador Responsable:	
Descripción:	
Observaciones:	

Figura 2. *Plantilla a usar para historias de usuario*

TAREAS DE INGENIERÍA

"Una historia de usuario se descompone en varias tareas de ingeniería, las cuales describen las actividades que se realizaran en cada historia de usuario, asimismo las tareas de ingeniería se vinculan más al desarrollador, ya que permite tener un acercamiento con el código." (Ferreira, 2013).

TAREA DE INGENIERÍA	
Numero de Tarea:	**Numero de Historia (Nro. y Nombre):**
Nombre de Tarea:	
Tipo de Tarea: Desarrollo/Corrección/ Mejora/ Otra (especificar)	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción:	

Figura 3. *Plantilla a usar para tareas de ingeniería*

PRUEBAS DE ACEPTACIÓN

"Las pruebas de aceptación son de vital importancia para el éxito de una iteración y el comienzo de la siguiente, con lo cual el cliente puede conocer el avance en el desarrollo del sistema y a los programadores lo que resta por hacer. Además, permite una retroalimentación para el desarrollo de las próximas historias de usuario a ser entregadas." (Jeffries, 2013).

PRUEBAS DE ACEPTACIÓN	
Código:	**N° Historia de Usuario:**
Historia de Usuario:	
Condiciones de Ejecución:	
Entrada/ Pasos de ejecución:	
Resultado Esperado:	
Evaluación de la Prueba:	

Figura 4. *Plantilla a usar para pruebas de aceptación*

TARJETAS CRC (CLASE-RESPONSABILIDAD COLABORACIÓN)

Las tarjetas CRC, permiten conocer que clases componen el sistema y la interacción entre ellas, se componen de tres apartados: Nombre de Clase, Responsabilidad y Colaboradores.

TARJETAS CRC	
Nombre de la Clase: Nombre de la clase al cual hace referencia la tarjeta.	
Responsabilidades: Atributos y operaciones de la clase.	**Colaboradores:** Clases que colaboran con la clase citada en la tarjeta.

Figura 5. *Plantilla a usar para tarjetas CRC*

2.2.2.2. MARCO DE TRABAJO SCRUM

Según Schwaber y Sutherland (2013) Scrum sirve como guía para aquellos desarrolladores que enfrenten problemas completos, entregando productos con valor en el desarrollo de software que nos impulsa a ser cada vez más adaptativos.

Para el autor Díaz (2009) Scrum es una metodología ágil, y en este marco de procesos de mejoras continuas, en donde el trabajar en desarrollo de software implica dar un valor al cliente y potenciar a los integrantes del proyecto para que puedan cumplir con los objetivos deseados, Scrum se sitúa como uno de las mejores metodologías para la gestión de proyectos.

El autor Sommerville (2011), cuenta que Scrum ya no se divide en iteraciones sino que ahora son llamados Sprint que tienen por objetivo un incremento del sistema, siendo periodos fijos, siendo Scrum una guía en la planeación y administración de proyectos software.

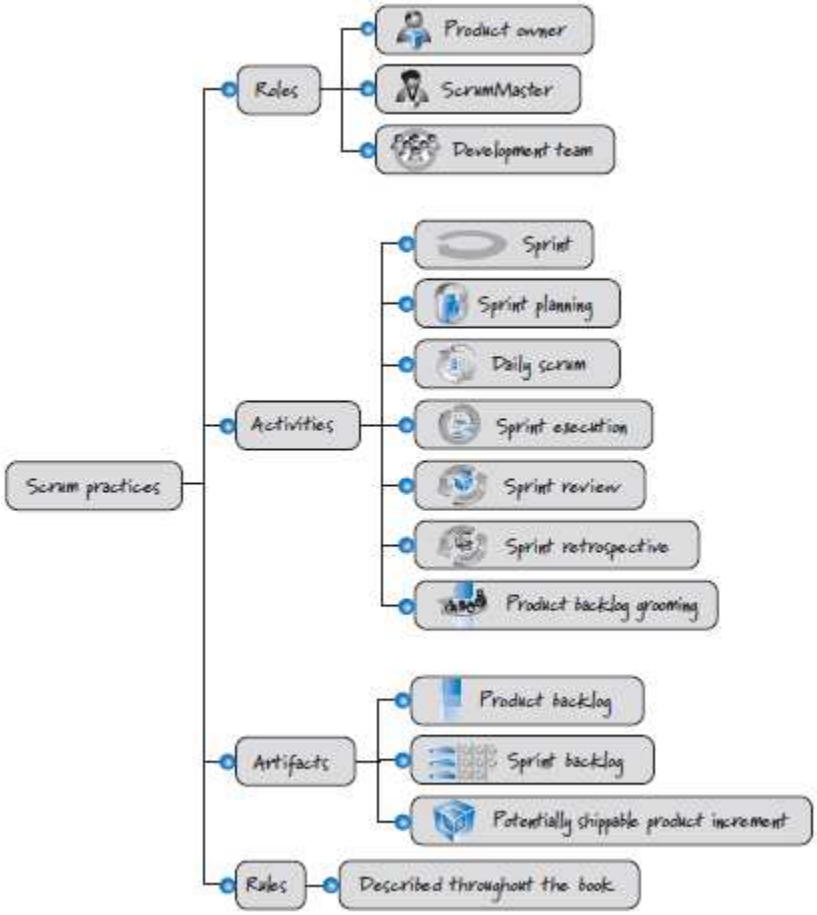


Figura 6. Prácticas de Scrum (Rubín, 2013)

A. ROLES SCRUM

a. El dueño de Producto (Product Owner)

"El propietario del producto es el punto central empoderado del liderazgo del producto. Él es la única autoridad responsable de decidir qué características y funcionalidades construir y el orden en el cual construirlos." El dueño del producto será quien vele por el éxito de este, siempre en constante contacto con el equipo para guiar en el uso de las prácticas de Scrum, aporta liderazgo y resolución de problemas. (Rubín, 2013).

Para los autores Schwaber y Sutherland (2013), es uno de los integrantes del equipo responsable de la gestión de la Lista del Producto (Product Backlog), de dar valor al producto y a cada miembro del equipo, sus métodos y técnicas varían dependiendo de lugar y de la persona en sí al igual de la institución.

b. El Scrum Master

Según Schwaber y Sutherland (2013), "es el responsable de asegurar que Scrum es entendido y adoptado. Los Scrum Masters hacen esto asegurándose de que el Equipo Scrum trabaja ajustándose a la teoría, prácticas y reglas de Scrum." Ayuda a todos los miembros internos y externos a entender lo que el equipo realiza, impulsa un ambiente de trabajo para crear valor en el producto que el equipo realiza.

Son los llamados "coach" que se encargan de que cada miembro del equipo entienda su trabajo utilizando las prácticas de Scrum, además también ayuda al cliente en la absolución de dudas, siendo un líder brinda el soporte necesario a los miembros. (Rubín, 2013).

c. El equipo de desarrollo (Development Team)

Para los autores Schwaber y Sutherland (2013), ellos son los profesionales que desempeñan el trabajo de entregar un incremento de producto "Terminado", que potencialmente se pueda poner en producción, al final de cada Sprint. Solo los miembros del Equipo de Desarrollo participan en la creación del incremento. Los Equipos de Desarrollo son estructurados y empoderados por la organización para organizar y gestionar su propio trabajo. La sinergia resultante optimiza la eficiencia y efectividad del Equipo de Desarrollo.

Los miembros que integran este equipo tienen diferentes profesiones, podemos encontrar ingenieros, tester, administradores, etc., todos con las habilidades y conocimientos para la elaboración de un software a exigencia de un propietario. (Rubín, 2013).

B. ACTIVIDADES

Figura 7. Actividades de Scrum (Deemer P. , Benefield, Larman, & Bas, 2011)

a. Planificación de Sprint (Sprint Planning)

"Es la planificación del trabajo a realizarse durante el Sprint. Este plan se crea mediante el trabajo colaborativo del equipo Scrum completo. La planificación de Sprint tiene un máximo de duración de ocho horas para un Sprint de un mes." El Scrum master es el encargado de hacer cumplir los tiempos de esta reunión, garantiza la asistencia de los miembros del equipo y el cliente. Hace cumplir con el propósito de la reunión. Si es un sprint corto, tendrá una duración menor. (Schwaber y Sutherland, 2013).

La reunión de Planificación, se da entre todos los miembros del equipo, en donde el dueño del producto presenta las historias de usuario que se encuentran en el Product Backlog y los otros miembros del equipo mediante preguntas aclaran sus dudas, para así por consensó tener la estimación del número de historias de usuario a realizar en un tiempo pactado, dividiendo las historias en tareas para el equipo de desarrollo. (Bahit, 2012).

En la planificación, se definen las funcionalidades que el equipo podrá realizar en el Sprint, sin exceder sus capacidades y los tiempos pactados, que serán entregados y presentados al finalizar el sprint antes de comenzar uno nuevo, es

por eso que el dueño del producto junto con el equipo de desarrollo se reúnen para pactar y tener un acuerdo. (Rubín, 2013).

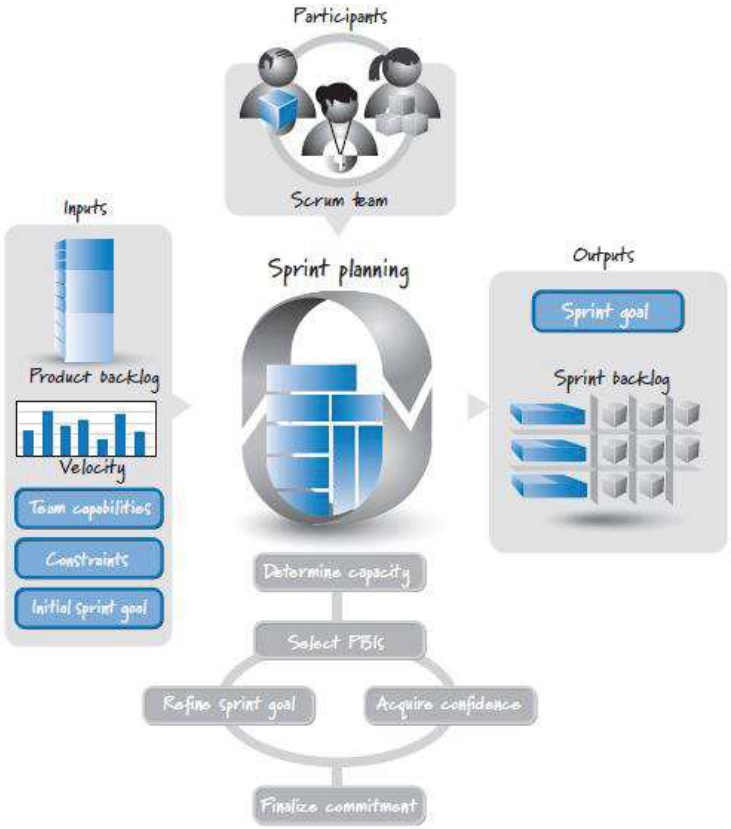


Figura 8. Actividades de la planificación del Sprint (Rubín, 2013)

b. Sprint

Schwaber y Sutherland (2013), es un "bloque de tiempo (time-box) de un mes o menos durante el cual se crea un incremento de producto terminado, utilizable y potencialmente desplegable. Es más conveniente si la duración de los Sprints es consistente a lo largo del esfuerzo de desarrollo. Cada nuevo Sprint comienza inmediatamente después de la finalización del Sprint previo."

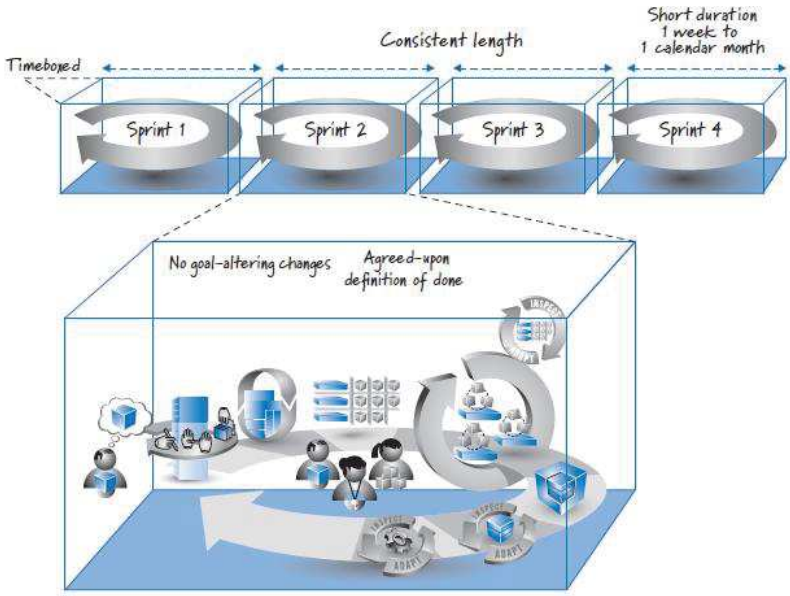


Figura 9. Los Sprints son el esqueleto del marco de Scrum (Rubín, 2013).

c. Daily Scrum

Reuniones diarias con una duración máxima de 15 minutos, para ubicar a los miembros del equipo en la situación actual del desarrollo y plantear las tareas que deberán hacer en las siguientes 24 horas, al ser diariamente se debe de completar las tareas previstas en la anterior reunión. (Schwaber y Sutherland, 2013).

"Las reuniones diarias para Scrum, son conversaciones de no más de 5-15 minutos, que el Scrum Master tendrá al comienzo de cada día, con cada miembro del equipo." Otra de las funciones del Scrum master es recabar información de lo que cada miembro, realizó el día anterior, revisar lo que hará en el día y sobretodo saber qué circunstancias y motivos surgieron para no cumplir las tareas y actividades, con el fin de resolver y que el equipo pueda continuar el trabajo previsto sin retrasos. (Bahit, 2012).

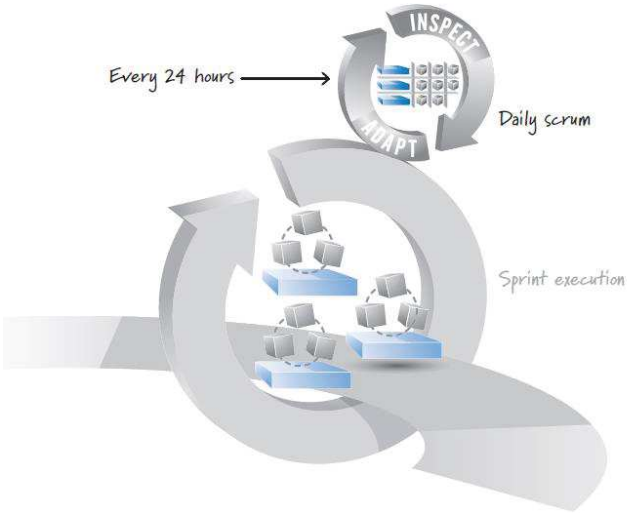


Figura 10. Scrum diario (Rubín, 2013)

d. Ejecución del Sprint (Sprint execution)

"Durante la ejecución del Sprint, los miembros del equipo de desarrollo realizan activamente trabajo creativo de diseño, construcción, integración y prueba de elementos de la lista de productos en incrementos de funcionalidad." Equipos auto organizados, con el único objetivo de desarrollar el proyecto, para cumplir con los tiempos y entregar software de calidad. Se planifican solos y se auto administran. (Rubin, 2013).

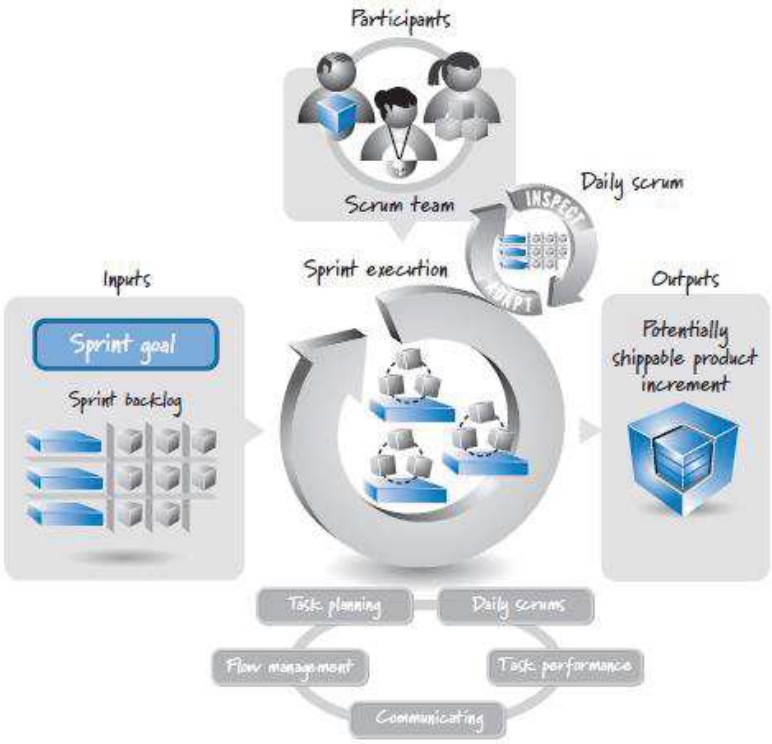


Figura 11. Actividad de ejecución de Sprint (Rubín, 2013)

e. Revisión de Sprint (Sprint review)

Como su nombre indica en esta reunión se celebra con todos aquellos interesados en la realización del producto y por tanto servirá como una reunión de retroalimentación y colaboración de participantes, es aquí donde se revisa que se hizo en este sprint, entonces se realiza al final de este, y servirá también para cambiar la pila si fuese necesario, con la colaboración de los asistentes, dialogar sobre qué otras actividades podrían surgir para dar aún más valor al producto. (Schwaber y Sutherland, 2013).

Al presentar en esta reunión de revisión las funcionalidades terminadas, los asistentes tendrán la oportunidad de escuchar la finalidad de dicho incremento, probarla y evaluarla para su futura implementación, si el dueño del producto no se encuentra satisfecho con sus necesidades tendrá la potestad de rechazar el incremento desarrollado, en caso contrario podrá aceptarla. Es llevada a cabo el último día del sprint, por la consecución de muchas actividades en esta reunión, no tiene una duración establecida, se tomará el tiempo que sea necesario. (Bahit, 2012).

El autor Rubin (2013) nos dice que en esta reunión se invita y realiza con todos los actores en la realización del software y hasta con miembros de otros equipos, cuyo objetivo es revisar la funcionalidad desarrollada, si cumple con las indicaciones del dueño del producto, y adaptar las características si es que el dueño del producto no está del todo satisfecho, todos los asistentes pueden dar sus opiniones sobre la funcionalidad y cómo poder mejorarla, compartir experiencias para poder mejorar en el siguiente sprint.

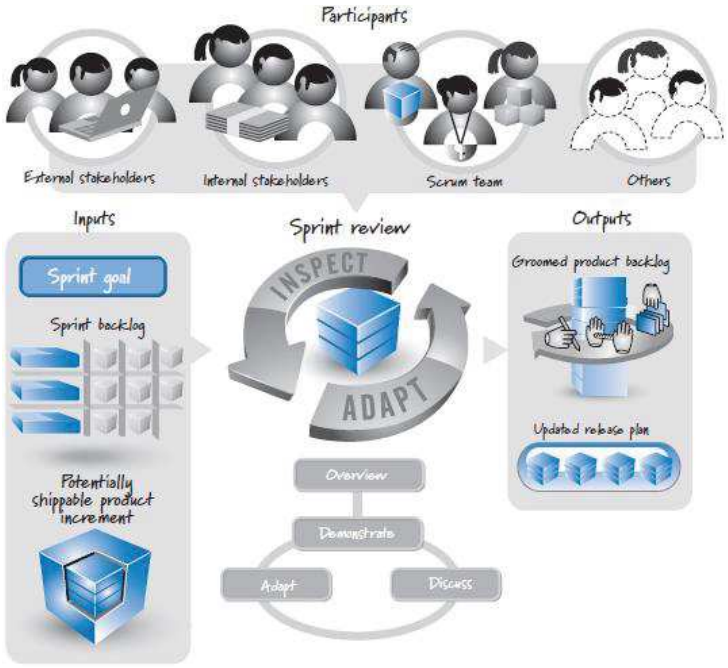


Figura 12. Actividad de revisión de Sprint (Rubín, 2013)

f. Retrospectiva de Sprint (Sprint Retrospective)

Esta reunión debe servir de autocrítica para el equipo de desarrolladores, para conocer sus fallas y sus aciertos, además de implementar un plan de mejora que será vistan en el otro sprint. Como quiera que fuese una reunión al terminar el sprint, el tiempo de duración serán de unas tres horas aproximadamente, podría decirse que sirve para preguntarse ¿Qué nos ha costado más en realizar? ¿Qué cosas hicimos bien? ¿Cómo podemos mejorar? (Schwaber y Sutherland, 2013).

Como última reunión del sprint sirve para poder plantear mejoras, establecer dificultades, y ver qué se puede mejorar para el siguiente sprint, saber si los resultados logrados fueron los suficientes para cumplir la funcionalidad prevista,

analizar los aspectos positivos para continuar con las prácticas, aspectos negativos que entorpecieron el desarrollo normal y de calidad del producto para establecer mejorías en el siguiente sprint. (Bahit, 2012).

Después de haber revisado el sprint y antes de planificar el siguiente, el equipo de desarrollo, el Scrum master se reúnen junto con el dueño del producto para verificar cuáles prácticas de Scrum realmente están funcionando y cuáles no, esta reunión tiene como objetivo adaptar e inspeccionar prácticas de Scrum al proyecto. "Al final de una retrospectiva de Sprint, el equipo Scrum debería tener identificado y comprometido un número práctico de acciones de mejora de procesos que será llevado a cabo por el equipo Scrum en el próximo Sprint." (Rubin, 2013).

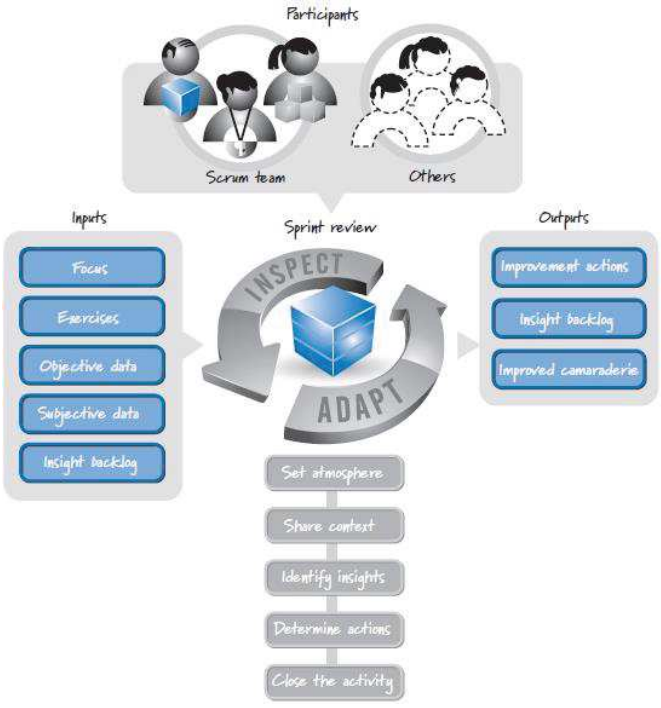


Figura 13. Actividad de retrospectiva de Sprint (Rubín, 2013)

C. ARTEFACTOS

a. Lista de Producto (Product backlog)

Lista que cuenta con todos los requerimientos que el dueño del producto brindó, con la cual se elabora el producto, cualquier cambio o duda se recurre a esta, tiene como características una descripción de lo que es y hace una característica del producto, un orden que determina del antes y después de que otro requisito irá, una estimación numérica que determina la prioridad de la misma y un valor para indicar la importancia que tiene. (Schwaber y Sutherland, 2013).

"Es una lista priorizada (u ordenada) de la funcionalidad deseada del producto. El propietario del producto colabora con las partes interesadas internas y externas para reunirse y definir los elementos de la lista de productos." Es una lista con orden nominal y jerarquizada de aquellos elementos requeridos por el dueño del producto, se realizan de acuerdo al orden de valor que aporta al cliente, pero que se entienda que son elementos que pueden ser cambiados, reemplazados o eliminados de acuerdo al entendimiento del equipo Scrum o por el mismo dueño del producto, tal como ellos vean posible, permitido porque el dinamismo y la flexibilidad del marco de trabajo de Scrum. (Rubin, 2013).

b. Sprint backlog

Al terminar un sprint y haber entregado un incremento de producto, y ya se piensa en el siguiente incremento, pero esto no sería posible sin la materia prima para todo sprint que son todos los componentes de la lista de productos que son elegidos para el sprint. (Schwaber y Sutherland, 2013).

c. Incremento del Producto

Al final de un sprint tendremos un incremento terminado, que cumple con las condiciones y especificaciones del dueño del producto, son incrementales, es decir, que también se llama incremento a aquellas funcionalidades terminadas de sprint anteriores, el dueño del producto tendrá la potestad de lanzarlos o esperar por el siguiente incremento. (Rubín, 2013).

2.2.2.3. MARCO DE TRABAJO KANBAN

Como otra metodología ágil tenemos a Kanban, que puede ser utilizada por desarrolladores de software en la medida que ayuden a gestionar mejorando los servicios, creativos esfuerzos y el diseño de productos físicos y de software. Eso puede caracterizarse como un método de "comenzar desde lo que haces ahora", un catalizador que promueve el dinamismo en una organización, en todo institución siempre hay personas temerosas y hasta reacias al cambio que tiene el fin de otorgar beneficios a la organización con el cumplimiento de sus metas (Carmichael y Andreson, 2015).

Ya sea una empresa de manufactura o una empresa de desarrollo, esta metodología nos ayuda con los materiales o información (o lo que sea), gestionando su flujo continuo en un proceso. No tener los materiales, ya sea una parte, un documento o información del cliente, lo cual causa retraso y residuos. Como hemos visto, el tener poco causa retraso pero de la otra forma, al tener mucho ya sea trabajo o materiales e información también causa retraso (residuos), es por ella que esta metodología nos enseña a gestionar el óptimo flujo de trabajo en cada proceso (Klipp, 2014).

A. VALORES

Para Carmichael y Anderson (2015) Como las otras metodologías estudiadas en este proyecto de investigación, también aquí encontramos valores, un conjunto de 9 valores, siendo el respeto el pilar de todos, que cada uno de ellos encapsula las teorías y prácticas de esta metodología.

a. Transparencia

El compartir información con los miembros mejora en gran medida los procesos otorgando valor a estos. Usando claro y sencillo el vocabulario es parte de este valor.

b. Balance

Valor que nos señala que debe de haber un equilibrio dentro de la empresa en cada aspecto del trabajo, evitando periodos largos.

c. Colaboración

Trabajando juntos. Kanban es formulado para enseñar a las personas a colaborar unas con otras, para cumplir un solo objetivo como empresa.

d. Enfoque en el cliente

Kanban al ser una metodología ágil, está enfocada a la satisfacción del cliente, siendo este un agente externo a la organización, pero cada producto obtenido debe de tener valor para ellos, el enfoque al cliente es primordial en Kanban, como empresa es importante dar valor a los productos solicitados.

e. Flujo

Comprender que el trabajo es una consecución de procesos y etapas, sean estas constantes o por partes, continuar con este flujo es esencial en Kanban.

f. Liderazgo

Las organizaciones al contar con personas que trabajan conjuntamente, necesitan alguien que guíe al grupo, con sus actitudes y aptitudes podrá inspirar al grupo con el fin de dar valor y mejorar los procesos, aun así siendo una organización jerárquica.

g. Comprensión

Principalmente al comprender el flujo de trabajo, las personas otorgan valor al producto, entonces es importante conocerse a sí mismo y conocer a la organización como tal.

h. Acuerdo

Contar con un ambiente laboral adecuado en donde se pueda convivir así se tenga opiniones o enfoques distintos, contribuye al cumplimiento de objetivos y al acuerdo de avanzar todos como un único grupo. Esto no es gestión por consenso, sino un co compromiso dinámico con la mejora.

i. Respeto

Valorar, comprender y mostrar consideración por personas. Al pie de esta lista, es la base de que los otros valores descansan.

B. OBJETIVOS

Para Richter (2011) los objetivos personales de Kanban son:

a. Al visualizar su flujo de trabajo, su trabajo aparecerá en su propio contexto que es fácil de comprender y fácil de reflexionar.

b. A través de la reflexión, comenzará a mejorar su flujo de trabajo y alcanzará un mayor valor de este esfuerzo.

c. Al limitar su trabajo en progreso, usted guiará su enfoque y logrará un rendimiento más alto.

C. AGENDAS

Para Carmichael y Anderson (2015) Kanban reconoce tres agendas basadas en necesidad organizacional:

a. La sostenibilidad

La agenda es sobre encontrar un sostenible ritmo y mejorando el enfoque.

b. El servicio Orientación Agenda

Enfoca atención al rendimiento y cliente satisfacción.

c. La supervivencia

La agenda está preocupada con mantenerse competitivo y adaptativo.

D. PILARES

Para Richter (2011) son 3 pilares personales de efectividad:

a. Importancia

- Aprende a seguir tu trabajo.
- Aprenda a priorizar su trabajo.
- Aprenda a respetar su propia priorización.

b. Atención

- Limitar el trabajo en progreso lo ayudará a mantener el enfoque.
- Aprender a manejar interrupciones externas.
- Aprender a manejar el retraso.
- Trabaja enfocado por 25 minutos y recompénsate con un freno de 5 minutos.

c. Valor

- Visualice su flujo de trabajo para aprender implícitamente sobre su flujo de valor.
- Limite el trabajo en progreso para ayudarlo implícitamente a seguir, aumentar su rendimiento y, por lo tanto, agregar más valor.
- Mapee su flujo de valor para visualizar etapas de agregar valor y evitando en la medida posible los cuellos de botella.
- Aprender sobre ciudadanos e integrarlos como parte de su personalidad.

E. PRACTICAS GENERALES

Carmichael y Anderson (2015) definen 6 de ellos:

a. Visualiza

Un tablero Kanban es una, aunque no la única, forma de comprender todo el trabajo, lo que se va a hacer y lo que se está haciendo, no entenderlo como un sistema de flujo, cumplir con lo acordado, y los límites de WIP deben mostrarse para controlar el trabajo del personal en cada etapa. Un tablero de pared, en pantallas electrónicas u otros medios: ayudan a ver, comprender en gran medida el trabajo global que se debe de hacer e identificar mejoras con el fin de entregar valor.

b. Limite el WIP

Introducir y respetar los límites de WIP cambia un pensamiento "push" en un pensamiento de "extracción", en donde los elementos nuevos no se inician hasta que el trabajo se complete (o en raras ocasiones, se aborta). Tener demasiado el trabajo parcialmente completo es derrochador y costoso y, crucialmente extiende los plazos de entrega, evitando que la organización sea receptiva a sus clientes y circunstancias cambiantes y a las oportunidades.

c. Administrar flujo

Contar con un flujo de trabajo, que sea entendible y predecible para la organización, en donde se minimice los tiempos y se entregue productos con máximo valor, son principios importantes en Kanban. En el día a día todo esto resulta ser más complicado aún, porque ya sea que tenemos que consultar y depender de otras áreas o hacer el control de los procesos con más dificultades encontrados, por lo que se deben de mejorar los procesos para evitar cuellos de botella.

d. Hacer políticas explícitas

Contar con un marco de trabajo simple y entendible, sencillos para el equipo, bien definidas pero modificables, visibles y pudiendo ser aplicadas por el personal, a tener una documentación extensa del marco de trabajo de la empresa, o contar con largos párrafos de misión y visión, terminología técnica y confusa en la explicación de los procesos, generan cuellos de botella, retrasos en la ejecución de

procedimientos y contribuye a que el flujo de trabajo se prolongue más de lo debido. Saber que "siempre aplicado" y "fácilmente cambiable" van juntos. Establecer límites de WIP, y luego nunca desafiar, cambiando o rompiendo los límites para ver si diferentes límites aplicados en otras circunstancias cambian los resultados, constituyendo una mala práctica de los principios.

e. Implementar bucles de retroalimentación

Aplicar reuniones de retroalimentación dentro de todo proceso de desarrollo es importante para la corrección de errores y mejora de procesos, una guía de retroalimentación se debe de aplicar en los siguientes puntos para obtener cambio positivo constante:

- Ordenar la estrategia
- Acuerdo operativo
- Administración y control de riesgos
- Evolución positiva y constante del servicio
- Reposición
- Fluir
- Dar producto de valor a los clientes

f. Mejorar como grupo, evolucionar empíricamente

Kanban tiene principios y filosofía de mejora constante. Debemos de evitar el querer transformar los procesos y mejorar pero hacia un enfoque ya pre definido. En contraste, Kanban comienza desde la organización tal como es ahora y utiliza el paradigma Lean flow (viendo trabajar como un flujo de valor) para perseguir continua e incremental mejora. Kanban aprovecha un proceso evolutivo para permitir un cambio beneficioso que puede ocurrir, evitando problemas que pueden resultar en la extinción de la organización. Una empresa no puede darse el lujo de no optar por no participar en la evolución: funciona para ellos o les sucede a ellos. Pero pueden optar por alentar que el cambio ocurra desde adentro, en lugar de encontrarlo, no puede responder a amenazas existenciales desde afuera. Kanban facilita esto.

F. ROLES

Kanban es y sigue siendo el método "comienza con lo que haces ahora", donde inicialmente nadie recibe nuevos roles, responsabilidades o títulos de trabajo. Por lo tanto, no hay roles obligatorios en Kanban y el método consiste en no crear nuevos puestos en la organización. Pero, dos roles vienen surgiendo del trabajo en campo y ahora se han incorporado dentro de la metodología. El objetivo de estos nuevos roles es lo que lo hace importante, en lugar de darle a alguien un título de trabajo, se ha visto útil pensar en estos roles como "sombreros" que identifican sus funciones: (Carmichael y Anderson, 2015).

A. El administrador de solicitudes de servicio (Service Request Manager), Al estar en relación con el cliente, tiene un amplio conocimiento de los requerimientos, por tanto ayuda en la organización de los elementos del trabajo en la reunión. Los nombres alternativos para el rol son Product Manager, Propietario del producto y Gerente de servicio.

B. El Service Delivery Manager es responsable del flujo de trabajar en la entrega de artículos seleccionados por los clientes y facilitar la reunión de Kanban y la planificación de la entrega. Alternativa de los nombres para este rol son: Flow Manager, Delivery Manager o incluso Flow Master.

G. FASES

Para el autor (Ballesteros, 2008), explica que tiene 4 fases para realizar el flujo de trabajo obteniendo un producto de valor con la aplicación adecuada de estos, siendo los siguientes:

- Fase 1: Preparar y concientizar a todos los miembros del equipo en las prácticas Kanban y todas las mejoras que trae.

- Fase 2: Utilizar las prácticas Kanban en todos los procesos que se ha identificado que tienen mayores problemas de hacer y resolver. Con el fin de tener, plantear soluciones y documentarlas para futuros procesos.

Se identifica al área de producción en donde la enseñanza debe de ser constante ya que son ellos los encargados de elaborar los productos.

- Fase 3: En esta fase luego de implementar en aquellos procesos con mayor dificultad, se continúa con los demás procesos y etapas, teniendo en cuenta las observaciones del personal para poder contribuir en la mejora de la metodología.

- Fase 4: En esta fase se debe de re evaluar los puntos vistos anteriormente, se debe de contar con revisiones periódicas de los procesos ejecutados con el fin de ordenar aquellos puntos o niveles con más dificultad. Se recomienda tener en cuenta:
  a. Todos los trabajos a realizar deben ser siguiendo la secuencia.
  b. Gracias a la comunicación, al encontrar problemas en los procesos, se debe de avisar a los jefes inmediatos superiores.

H. ARTEFACTOS

Según (Kanbanize, 2020) son:

a. "El tablero Kanban es la herramienta para mapear y visualizar su flujo de trabajo y uno de los componentes claves del método Kanban." Se trata de una pizarra limpia, blanca o con una amplia área para trabajar, dividida en columnas y filas, que representan una fase del proceso y las filas como las tareas y actividades.

El tablero Kanban se organiza de manera constante con cada tarea o actividad se colocará una nueva tarjeta Kanban, que será ubicada en la columna "Por hacer".

Hoy en día la tecnología avanza a pasos agigantados y surge la pregunta de cómo trabajar con equipos remotos, entonces llegan soluciones como las tarjetas digitales, pudiendo acceder a ellas a través del internet sin importar en donde se trabaje, practicidad y accesibilidad son sus características.

b. Cómo representar las tareas o actividades del proyecto en el tablero Kanban, encontramos así a las tarjetas Kanban que son claves para indicar lo que debe de realizarse.

Todas las tarjetas Kanban siempre contienen como características que permitirán identificar la información y el estado de cada una, como una descripción, tiempo de vida, fecha programada, entre otras, y es así como todos los integrantes del equipo se comunican entre ellos.

Entonces el objetivo de una tarjeta Kanban es indicar todo el proceso de una tarea o actividad, desde que entra al tablero Kanban hasta donde ya se considera que están realizadas y en este transcurso de tiempo, estas tarjetas ayudan al equipo de trabajo:

- Comunican información para el equipo.
- Al tener cada tarjeta sus características, ya se necesitan menos reuniones.
- Al ser vistas por todas las personas, existe una mayor transparencia del trabajo.

Exceder el número de tarjetas Kanban en una columna, retrasará el trabajo al momento de desarrollarlas, por eso, éstas deben de tener una cantidad adecuada para aminorar problemas y hacer cambios de contexto.

2.2.2.4. PROGRAMACIÓN EXTREMA SOBRE SCRUM

"La metodología de desarrollo ágil XP y el marco de trabajo Scrum se complementan entre sí. Del lado de XP, se destacan las prácticas, valores y el ciclo de vida que esta metodología propone, misma que se compone de seis fases: Exploración, Planeación, Diseño, Codificación, Pruebas del Proyecto." Xp y Scrum complementan bien en artefactos, procesos y eventos, por ejemplo en disminuir la documentación excesiva, integrar al cliente en el equipo de trabajo. Cubriendo así las necesidades de un proyecto software, teniendo buena gestión con calidad de código. (Carrasco, Ocampo, Ulloa, Azcona, 2019).

Al usar las metodologías agiles se busca la integración de al menos dos con el fin de al enfocarse a campos distintos del desarrollo se complementar mejor, una enfocada a las práctica de desarrollo y la otra a prácticas de organización y gestión del trabajo. (Kniberg, 2015).

A. SOFTWARE ITERATIVO E INCREMENTAL

Los autores explican "que la aplicación se divide en pequeños proyectos, los cuales incorporan una parte de las especificaciones, y el desarrollo de la misma es una iteración que va incrementando la funcionalidad del sistema de manera progresiva." (Silva, Barrera, Arroyave y Pineda, 2007).

Consiste en entregar al cliente funcionalidades incrementales y constantes con el fin de que pueda evaluar la realización de los mismos, para brindar retroalimentación al equipo del trabajo en las futuras actividades a venir, cumpliendo con dar valor al producto estas entregas deben ser en periodos de tiempo cortos. (Pressman, 2010).

Los requerimientos funcionales del producto, nos dice el autor, "son fragmentados en lo que se denomina Historias de Usuario y, basándose en la prioridad del cliente y el esfuerzo disponible para el desarrollo, se hace una selección de historias de usuario a ser desarrolladas en un periodo de tiempo fijo y al finalizar dicho periodo, las historias de usuario habrán sido convertidas en software que puede utilizarse y, por tanto, se entrega al cliente". Este ciclo, es continuo (al finalizar un ciclo comienza el siguiente) y esto, genera una entrega rápida y continuada al cliente (Bahit, 2012).

B. PRUEBAS UNITARIAS CONTINUAS

Para los autores (Tuya, Ramos, Dolado, 2007), dice que "constituyen el primer paso para detectar errores en el código, pues se centran en la menor unidad de diseño del software: el módulo – por ejemplo, un método de una clase o una clase A raíz de los entregables continuos y constantes, estas funcionalidades necesitan pasar por pruebas continuas para garantizar código limpio, coherente en estructura, en integridad y comportamientos, son testeadas por un programador

encargado de desarrollar estas pruebas, son hechas antes de que el módulo desarrollado, se integre al sistema.

Para el autor (Bahit, 2012), el liberar funcionalidades continuas exige que este producto pase por pruebas unitarias constantes, antes de poder integrarlas al sistema, garantizando así que tendremos código de calidad, estas pruebas son sencillas y simples que deben e comprobar que la funcionalidad cumpla de manera mínima lo que se acordó.

C. RE FACTORIZACIÓN DEL CÓDIGO

Como producto de la liberación de software utilizable constante se genera código, que puede ser nuevamente utilizado, es donde entra la re factorización que va a consistir en cambiar y optimizar la estructura interna del código, sin alterar su finalidad externa y sin perder funcionalidad. (Abrahamsson, Salo, Ronkainen y Warsta, 2002).

Es el proceso de cambiar la estructura de código, reformulándolo, sin cambiar su significado o comportamiento. Está acostumbrado a mejorar la calidad del código, combatir la inevitable entropía del software y facilitar agregando nuevas características (Shore y Warden, 2008).

D. CORRECCIÓN DE ERRORES Y CÓDIGO COMPARTIDO

Durante el transcurso de la elaboración de software se van a encontrar bugs, que van a generar que las funcionalidades tengan errores, si se les corrige y documenta lo más antes posible, se evitarán cometerlos después. (Jeffries, Anderson y Hendrickson, 2001).

La propiedad colectiva del código evita que los miembros no compartan el código con otros miembros, en un proyecto todos conocen todo ya que desarrollan el mismo sistema. “existen programadores dueños de un código y cuando surge un problema, nadie más que él puede resolverlo, puesto que el resto del equipo, desconoce cómo fue hecho y cuáles han sido sus requerimientos a lo largo del desarrollo.” (Bahit, 2012).

2.2.2.5. SCRUMBAN

Para (Pérez, 2011) sostiene que, "Scrumban (o Scrum-ban) es una metodología derivada de los enfoques Scrum y Kanban. Esta metodología, por cierto, híbrida, contempla componentes y conceptos de ambas que se complementan entre sí, para lograr una mejor optimización del proceso de desarrollo."

Este término es usado en el trabajo de investigación de (Ladas, 2008), "Scrumban-Essays on Kanban System for Lean Software Development".

Al integrar y complementar el marco de trabajo de Scrum y Kanban, permite utilizar prácticas de ambas metodologías, cosa que al estar separados no se podría dar, otorgando al proyecto conceptos de programación y organización. Este enfoque de Scrumban puede ser dado en dos distintas medidas:

- Se tiene por una parte que el trabajo es más enfocado a Kanban incorporando características y prácticas de Scrum, y

- Por otra parte cuando el proyecto está más enfocado a Scrum y se le incorpora prácticas de Kanban como el tablero y tarjetas Kanban que ayudarán a tener una visión global de todo el proyecto.

Este complemento de metodologías parte de tener una base simple y gracias a que las prácticas son flexibles se llega cada vez a que sea más compleja pero manejable.

A. CARACTERÍSTICAS DE SCRUMBAN

Para el autor (Ahmad, 2014) Scrumban tiene las siguientes:

a. Visualizar el flujo de trabajo

Como parte del uso de las políticas explícitas, tenemos el uso de la tabla Kanban para este marco de trabajo que es una herramienta que ayuda en gran medida a visualizar cada una de las tareas y actividades, conocidas como historias de usuarios, nos ayuda también a controlar el número de actividades por columna y verificar el nivel de progreso de una historia de usuario para identificar aquellas que implican un mayor esfuerzo en realizarlas.

b. Cola de trabajo

Como producto del diálogo con los clientes se obtienen las historias de usuario, estas son valorizadas y estimadas, para ser seleccionadas para un sprint que se terminará en el tiempo fijo estimado en cumplimiento con el cliente, si trabajásemos por metodologías separadas, estas historias de usuario no se podrían modificar al haber iniciado un sprint pero en este marco de trabajo híbrido existen las colas der trabajo que van a permitir el cambio de las historias de usuario en caliente sin significar cambios que afecten la estructura o su comportamiento.

c. Limitar el trabajo en progreso (WIP)

El tener excesivas historias de usuario en cualquiera de las columnas del tablero de Scrumban provocará estrés en el equipo, pudiendo apresurar a los miembros a cumplir con el desarrollo, entonces tener un límite de tareas por realizar previamente definidas viendo la capacidad del equipo, contribuye en acelerar el flujo, evita cometer errores y trabajar a un ritmo sostenido, el conservar un ambiente agradable de trabajo, siempre contribuye a obtener un producto de mejor calidad.

d. Reglas explícitas

En Scrumban los equipos se organizan ellos mismo, permitiendo el uso de reglas explícitas, en la práctica esto puede cambiar al ser un proceso dinámico, lo que se dice y lo que se hace no siempre van de la mano. Por lo tanto este marco de trabajo posibilita la toma de decisiones de manera rápida sin invertir grandes tiempos y recursos del proyecto. Estas políticas explícitas ayudan a los miembros del equipo, en la toma de decisiones rápidas frente al innumerable sinfín de situaciones que se pudiesen presentar al realizar el proyecto.

B. REUNIONES EN SCRUMBAN

En este marco híbrido se mantienen las reuniones de ambas metodologías, cambiando solamente la duración entre reuniones que tiene cada una.

a. Reuniones de Planificación

"A diferencia de Scrum, Scrumban tiene reuniones de planificación más cortas, con el fin de actualizar el backlog cuando sea necesario. El equipo siempre debe planificar para el período más corto por delante." No es conveniente tener reuniones de larga duración ya que el marco de trabajo supone que las prioridades sean cambiantes, por lo tanto se debe ser coherente con la duración de estas reuniones. (Ladas, 2008).

b. Reunión Stand-up (diaria)

Reuniones diarias con el fin de gestionar y coordinar las actividades y la exposición de dificultades durante el trabajo del día anterior. Tendrá como objetivo la retroalimentación de sus partes, refrescar los objetivos del sprint, comentar problemas y errores surgidos durante del día, documentarlos en la medida posible para tener soluciones futuras automatizadas.

c. Reunión de Revisión de Sprint

Las reuniones de revisión son exactamente iguales a las del marco Scrum. Teniendo la misma duración y tratando los mismos puntos.

d. Reunión de Retrospectiva

Reuniones para ver hacia atrás, cada equipo tendrá su forma de realizarlas, pero servirá para identificar los problemas que se presentaron con el fin de prever y dar solución si llegasen a presentarse después. Al tener una visión de lo que se hizo, se puede ver a futuro, lo que se puede hacer y evitar en el siguiente sprint para cumplir con los plazos acordados.

2.3. HERRAMIENTAS

2.3.1. PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

El autor Pérez (2014), cuenta que la programación orientada a objetos es una guía para programar software mediante objetos y clases que cuentan con sus propias prácticas de desarrollo, al usar métodos que serán comunes entre las clases se pueden aplicar características como herencia, encapsulamiento, polimorfismo.

Para el autor Martínez (2016), este modelo trabaja con clases que están compuestas por objetos como un modelo abstracto que tienen características y atributos que pueden ser agrupados al tener similitudes poniendo énfasis en los objetos que en mismo flujo de trabajo.

"La programación orientada a objetos es un importante conjunto de técnicas que pueden utilizarse para hacer el desarrollo de programas más eficientes, a la par que mejora la fiabilidad de los programas de computadora." Como parte importante de la programación orientada a objetos, supone el trabajo con objetos, entender su comportamiento, características y cómo estos se relacionan con otros objetos del sistema, básicamente programación orientada a objetos estudia un conjunto de objetos que trabajan y se relacionan entre sí, que se organizan en clases. (Joyanes, 2005).

2.3.2. SISTEMA GESTOR DE BASE DE DATOS

Un SGBD son programas de manipulación y consulta de datos que sirven de intermediario e interfaz para el usuario y toda la colección de datos de una base de datos, evita la mala manipulación de los datos. (Castillo, 2011).

Son un conjunto de datos interrelacionas que cuentan con programas para el acceso a estos, estos datos se organizan y se conocen como base de datos que es información importante y vital para las empresas. “El objetivo principal de SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto practica como eficiente.” (Silberschatz et al., 2006).

“Un sistema de administración de base de datos (DBMS) es el software que permite a una organización centralizar los datos, administrar eficientemente y proporcionar, mediante los programas de aplicación, el acceso a los datos almacenados.” Sirve como interfaz para el usuario entre programas y datos físicos. (Laudon, K. y Laudon, J., 2008).

2.4. POBLACIÓN

Según Chávez (2007), la población “es el universo de estudio de la investigación, sobre el cual se pretende generalizar los resultados, constituida por características o estratos que le permiten distinguir los sujetos unos de otros”.

Para el autor Monroy (2008), viene a ser todos los elementos que integran el universo. Al realizar una investigación se analiza y estudia una variable que es del interés para el investigador.

Córdova (2003), afirma que vienen a ser características que se puedan mediar en personas, objetos, materiales, etc., y a cada elemento del universo se le conoce como unidad estadística o elemental.

2.5. MUESTRA

Para el autor Monroy (2008), cuando se ejecuta un proyecto de investigación, resulta muy complicado estudiar la población en su totalidad porque supondría invertir recursos y tiempo para estudiar cada uno de los elementos que la compone, es porque sólo se estudia una parte de la población llamada muestra.

El autor Córdova (2003), dice que "se denomina muestra a una parte de la población seleccionada de acuerdo con un plan o regla, con el fin de obtener información acerca de la población de la cual proviene. La muestra debe ser seleccionada de manera que sea representativa de la población." Puede ser seleccionada al azar o sea cada elemento de la población tiene la misma posibilidad de ser seleccionada.

CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1. TIPO Y NIVEL DE INVESTIGACIÓN

3.1.1. TIPO DE INVESTIGACIÓN

"En los estudios observacionales no existe intervención de ningún tipo por parte del investigador, de manera que los datos observados y la información consignado refleja la evolución natural de los eventos." (Supo, 2012). Este trabajo de investigación es de **tipo observacional.**

"Los estudios retrospectivos utilizan datos que se obtienen de registros preexistentes, datos que provienen de mediciones en donde el investigador no tuvo participación alguna. A este tipo de información se le suele llamar datos secundarios." (Supo, 2012). Este trabajo de investigación es de **tipo retrospectivo.**

"En un estudio transversal todas las variables (incluyendo la variable de estudio) son medidas en una sola ocasión bajo esta condición, si realizamos comparaciones entre estas mediciones se les suele llamar entre muestras independientes, aunque el nombre correcto sería entre grupos independientes." (Supo, 2012). Este trabajo de investigación es de **tipo transversal.**

3.1.2. NIVEL DE INVESTIGACIÓN

"Los estudios descriptivos buscan especificar las propiedades, las características y los perfiles de personas, grupos, comunidades, procesos, objetos o cualquier otro fenómeno que se someta a un análisis. Es decir, únicamente pretenden medir o recoger información de manera independiente o conjunta sobre los conceptos o las variables a las que se refieren, esto es, su objetivo no es indicar como se

relacionan estas.” (Hernández, et al, 2014). Este trabajo de investigación es de **nivel descriptivo.**

3.1.3. DISEÑO DE LA INVESTIGACIÓN

“Los diseños no experimentales son estudios que se realizan **sin manipular deliberadamente variables**. Es decir, se trata de estudios en los que no hacemos variar en forma intencional las variables independientes para ver su efecto sobre otras variables. Lo que hacemos en la investigación no experimental es observar fenómenos tal como se dan en su contexto natural, para analizarlos.” (Hernández et al., 2014). Este trabajo de investigación es de diseño **no experimental.**

“Los diseños de investigación transversal recolectan datos en un solo momento, en un tiempo único. Su propósito es describir variables y analizar su incidencia e interrelación en un momento dado. Es como tomar una fotografía de algo que sucede.” (Hernández et al., 2014). Este trabajo de investigación es de diseño **transversal.**

Como en este trabajo de investigación no se van a manipular las variables y también los datos usados se midieron una sola vez, siendo por tanto de diseño no experimental y transversal.

3.2. POBLACIÓN Y MUESTRA

3.2.1. POBLACIÓN

La población de estudio estará compuesta por todos los modelos de gestión de desarrollo de software ágil.

3.2.2. MUESTRA

Se tomó los modelos de gestión de desarrollo de software ágil a Scrum y Kanban.

3.3. VARIABLES E INDICADORES

3.3.1. DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

VARIABLE DE ESTUDIO

Modelo de gestión de desarrollo de software ágil. En el ámbito de desarrollo de software intervienen un conjunto de prácticas como la gestión, ya que sin una organización y administración del personal, de las prácticas no se alcanza el objetivo, también la planificación y estimación de costos y tiempos para cumplir con los compromisos de costos y tiempos fijados.

INDICADORES DE LA VARIABLE

Scrum. Como una de las metodologías usadas en esta investigación, este marco guía nos ofrece la posibilidad de desarrollar un producto a través de periodos o tiempos establecidos llamados sprint, que se constituyen de historias de usuarios jerárquicas, ofreciendo en todo este proceso una adecuada administración para el cumplimiento de objetivos.

Kanban. Otra metodología usada en esta investigación que brinda una guía de cómo implementar todo un flujo de trabajo a través de elementos que brindan transparencia y control para todos los miembros del equipo.

VARIABLE ESTUDIO

Programación Extrema. Siendo la metodología de desarrollo más usadas por sus altas ventajas y beneficios como lo son sus prácticas enfocadas en la programación, teniendo disposición del cliente 24/7 y el incremento de una funcionalidad en cada iteración.

INDICADORES DE LA VARIABLE

Software iterativo e incremental. Es la entrega de prototipos ejecutables en periodos cortos de tiempo. Este enfoque permite aprovechar la retroalimentación entre el cliente y el equipo de desarrolladores, por tanto, se incrementa la funcionalidad del sistema de manera progresiva.

Software a través de las pruebas unitarias. Consiste en identificar mala praxis en la realización del código a medida que se termina el desarrollo de cada incremento, se realiza de manera individual a cada módulo.

Software a través de re factorización de código. Como parte del mantenimiento del código, este debe de estar en continuo mejoramiento, quitando código obsoleto pero sin cambiar la funcionalidad del mismo, obtenemos así, al finalizar un producto de calidad.

Software a través de corrección de errores y código compartido. Es un proceso continuo que implica la prevención de errores, conjuntamente con la corrección in situ cuando se producen, mejora así la calidad del software.
Nadie es dueño de nada y mucho menos del código elaborado, de tal manera que al involucrar en el desarrollo a todos los miembros, cada uno de ellos tendrá la capacidad de entender y corregir problemas de código. Esto implica comentar el código que se hace porque si alguien renuncia o es despedido sólo él conocería la funcionalidad de lo que desarrolló.

3.3.2. DEFINICIÓN OPERACIONAL DE LAS VARIABLES DE ESTUDIO

VARIABLE DE ESTUDIO

X: Modelo de gestión de desarrollo de software ágil.

INDICADORES

X1: Scrum.

X2: Kanban.

VARIABLE DE ESTUDIO

Y: Programación Extrema.

INDICADORES

Y1: Iterativo e incremental.

Y2: Pruebas unitarias continúas.

Y3: Refactorización del código.

Y4: Corrección de errores y código compartido.

3.4. TÉCNICAS E INSTRUMENTOS PARA RECOLECTAR INFORMACIÓN

3.4.1. TÉCNICAS

- Análisis documental.

3.4.2. INSTRUMENTOS

- Registro de ficha.

3.5. HERRAMIENTAS PARA EL TRATAMIENTO DE DATOS E INFORMACIÓN

Son seleccionadas tomando en cuenta la ayuda que nos brindará en el tratamiento de los datos e información obtenidos. En la siguiente figura se muestran las herramientas usadas:

SOFTWARE	FABRICANTE	SERVICIO
Windows 10	Microsoft Corporation	Versión más reciente de este S.O.
Netbeans 8.2	Sun Microsystems	Orientada a objetos, desarrollo de software por módulos.
Mysql	MySQL AB, Sun Microsystems y Oracle Corporation	Sistema de gestión de bases de datos relacionales de código abierto con un modelo cliente-servidor.
Jira	Atlassian	Jira Software

Figura 14. *Herramientas para el tratamiento de datos e información*

3.6. PROPUESTA DE MODELO DE GESTIÓN EN FUNCIÓN A LAS METODOLOGÍAS APLICADAS

ESTRATEGIA DE INTEGRACIÓN

El reto en este trabajo de investigación ha sido integrar estas tres metodologías, combinar y diferencias sus prácticas y enfoques, con el objetivo de brindar el mejor marco de desarrollo, administración y gestión del proyecto software, que no se podría obtener al trabajar individualmente con cada metodología separada.

Con Xp tenemos un enfoque de prácticas de programación y consecución de pruebas unitarias, con Scrum, un enfoque de prácticas de gestión y organización y Kanban más enfocado en gestionar de manera general como se van completando tareas y en la mejora continua, adaptándose como la base de esta propuesta de modelo de gestión de trabajo.

Entonces surge la interrogante: ¿Qué ventajas competitivas obtendré al integrar estas tres metodologías? Al liberar software utilizable de manera continua podrá utilizarse sin que se deba de esperar a que acabe el proyecto, entonces los dueños del producto, tendrán a sus clientes más satisfechos lo que se traduce en ingresos para esa empresa, diferenciación con otras empresas del mismo rubro, se han integrado las metodologías en los siguientes gráficos:

	XP	SCRUM	KANBAN	NOTA
ROLES	Cliente	Product Owner	Service Request Manager	Similares funciones
	Encargado de seguimiento	Scrum Master	Service Delivery Manager	Similares funciones
	Programador	Scrum Team		Similares funciones
	Encargado de pruebas			Adoptado en esta propuesta de modelo de gestión de trabajo ágil

Figura 15. *Modelo de integración de las metodologías Xp, Scrum y Kanban (Roles)*

	XP	SCRUM	KANBAN	NOTA
ACTIVIDAD Y PROCESOS	Planeación	Product Backlog (Pila del producto)	Concientización de la metodología	Similares funciones
	Análisis	Sprint Planning		Complementar con el análisis de requerimientos
	Diseño			
	Codificación	Sprint	Implementar priorizando componentes con más dificultades	Diseño simple fácilmente entendible que ayudará en la comunicación y desarrollo
	Pruebas			
	Pruebas unitarias Pruebas de aceptación Pruebas de integración		Implementar el resto de componentes	Se adoptará en esta propuesta de modelo de gestión de trabajo ágil
		Scrum diario	Revisión del Sistema Kanban	Adoptar en esta propuesta de modelo de gestión de trabajo ágil
		Sprint Review Meeting		
		Retrospectiva		
		Re Release Planning Meeting		

Figura 16. *Modelo de integración de las metodologías Xp, Scrum y Kanban (Actividades y Procesos)*

	XP	SCRUM	KANBAN	NOTA
ARTEFACTOS	Historias de usuario	Product Backlog (Pila del producto)		Combinadas unas con otras para mejorar las especificaciones del cliente
	Tarjetas CRC	Sprint Backlog		
	Task Cards			
		Burn down chart	Tablero Kanban	
		Burn up chart	Cartas Kanban	

Figura 17. *Modelo de integración de las metodologías Xp, Scrum y Kanban (Artefactos)*

	XP	SCRUM	KANBAN	NOTA
VALORES	Comunicación	Comunicación	Transparencia	Similitud conceptual
			Colaboración	
			Entendimiento	
	Retroalimentación	Retroalimentación	Equilibrio	
			Enfoque al cliente	
	Sencillez		Flujo	
	Valentía	Responsabilidad		
	Respeto		Liderazgo	
			Acuerdo	
			Respeto	

Figura 18. *Modelo de integración de las metodologías Xp, Scrum y Kanban (Valores)*

	XP	SCRUM	KANBAN	NOTA
PRACTICAS	Cliente in situ	Cliente presente en todo el proceso		Combinar unas con otras para poder mejorar pruebas, estandarizar código, etc.
	Semana de 40 horas	Iteración de 2 a 4 semanas	Visualizar el flujo de trabajo	
	Metáfora		Gestionar el flujo	
	Diseño simple		Políticas explícitas	
	Re factoring		Retroalimentación	
	Programación en parejas		Eliminar interrupciones	
	Liberaciones cortas	Liberaciones cortas		
	Pruebas			
	Código estándar			
	Integración continua	Integración continua		
	Propiedad colectiva			
	Juego de planificación			

Figura 19. *Modelo de integración de las metodologías Xp, Scrum y Kanban (Practicas)*

De esta manera la propuesta de modelo de gestión de trabajo ágil "XP + Scrum + Kanban", tendríamos nuevas figuras:

ROLES	Product Owner
	Scrum Master
	Scrum Team
	Encargado de pruebas

Figura 20. *Propuesta de modelo de gestión de trabajo ágil "XP + Scrum + Kanban" - Roles*

<table>
<tr><td rowspan="11">ACTIVIDADES Y PROCESOS</td><td colspan="3">Planeación</td><td rowspan="3">Concientizar nueva metodología</td></tr>
<tr><td colspan="3">Análisis</td></tr>
<tr><td rowspan="2">Diseño</td><td>Políticas explícitas</td><td rowspan="2">Sprint Planning</td></tr>
<tr><td>Diseño simple</td></tr>
<tr><td colspan="2">Codificación</td><td rowspan="4">Sprint</td><td rowspan="4">Implementación de todos los componentes</td></tr>
<tr><td rowspan="3">Pruebas</td><td>Pruebas unitarias</td></tr>
<tr><td>Pruebas de aceptación</td></tr>
<tr><td>Pruebas de integración</td></tr>
<tr><td colspan="3">Scrum diario</td><td rowspan="4">Revisión del sistema</td></tr>
<tr><td colspan="3">Sprint Review Meeting</td></tr>
<tr><td colspan="3">Retrospectiva</td></tr>
<tr><td colspan="3">Re Release Planning Meeting</td></tr>
</table>

Figura 21. *Propuesta de modelo de gestión de trabajo ágil "XP + Scrum + Kanban" - Actividades y Procesos*

<table>
<tr><td rowspan="5">ARTEFACTOS</td><td>Historias de Usuario</td><td colspan="2">Product Backlog</td><td rowspan="2">Tablero Kanban</td></tr>
<tr><td>Tarjetas CRC</td><td colspan="2" rowspan="2">Sprint Backlog</td></tr>
<tr><td>Task Cards</td><td rowspan="3">Cartas Kanban</td></tr>
<tr><td colspan="3">Burndown chart</td></tr>
<tr><td colspan="3">Burn up chart</td></tr>
</table>

Figura 22. *Propuesta de modelo de gestión de trabajo ágil "XP + Scrum + Kanban" - Artefactos*

VALORES	Comunicación
	Enfoque al cliente
	Sencillez
	Responsabilidad
	Retroalimentación

Figura 23. *Propuesta de modelo de gestión de trabajo ágil "XP + Scrum + Kanban" - Valores*

PRÁCTICAS TÉCNICAS	Iteración de 2 a 4 semanas	Limitar el WIP
	Visualizar el flujo de trabajo	
	Liberaciones cortas	
	Programación en parejas	
	Re factoring	
	Código estándar	
	Integración continua	
	Propiedad colectiva	

Figura 24. *Propuesta de modelo de gestión de trabajo ágil "XP + Scrum + Kanban" - Prácticas Técnicas*

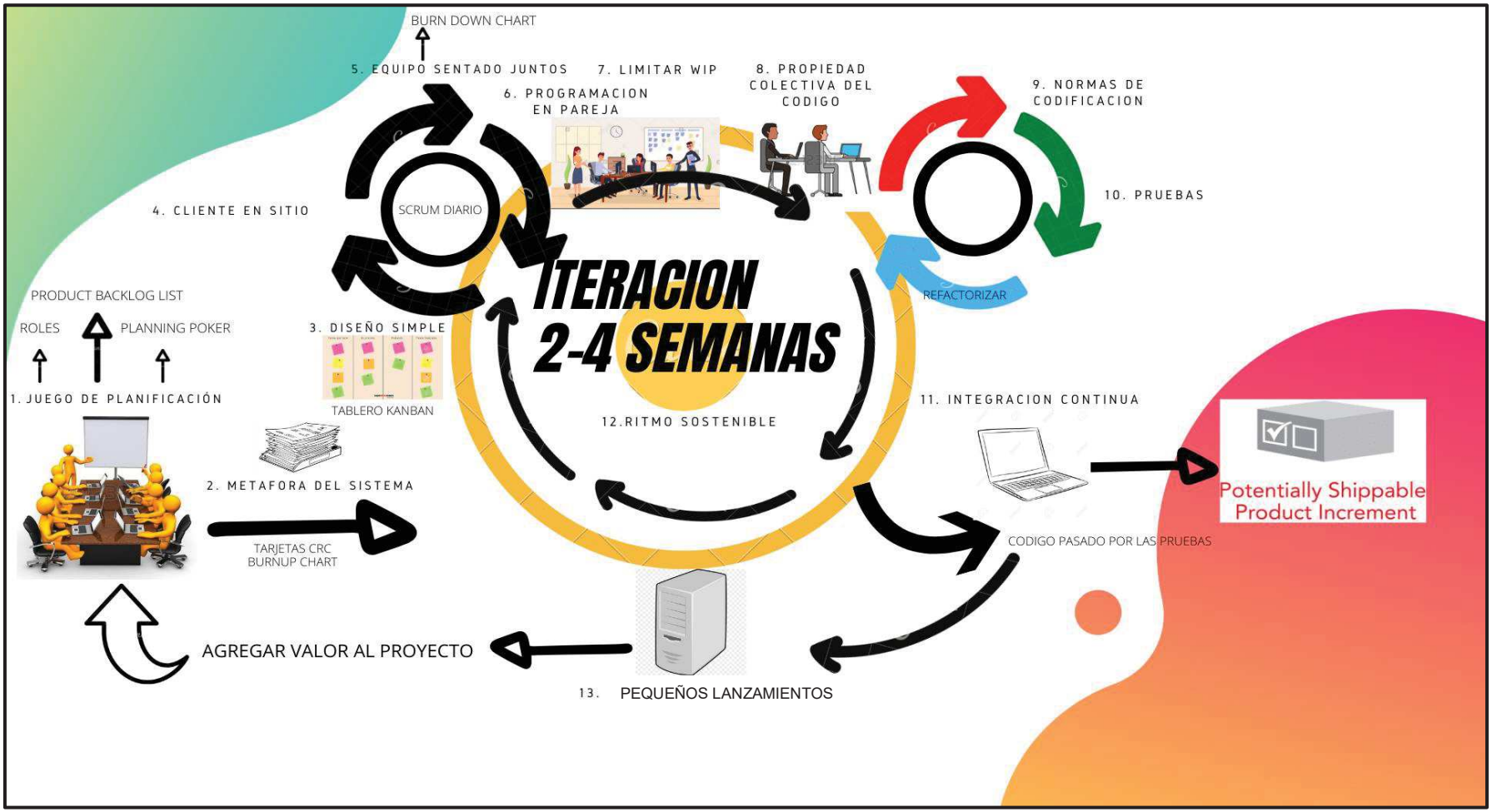


Figura 25. Marco de trabajo “XP + Scrum + Kanban”

1. Juego de planificación

Se empiezan a elaborar las historias de usuario como parte del dialogo con el cliente, también los periodos y fechas de presentación de un incremento. Se estima y valora las historias de usuario para seleccionarlas para la iteración, estas se descomponen en tareas por los programadores.

Se define el trabajo que se va a realizar en el proyecto, se comienzan a enumerar las funciones generales y específicas de cada módulo, estas tendrás mayor importancia y serán desarrolladas de manera jerárquica, con un plazo previamente acordado de liberación. Esta primera etapa es considerada como un cronograma del proyecto en donde se empezarán a valorar los requerimientos recogidos. Entender el producto como un todo, ayuda siempre para comprender lo que se hará. En esta primera parte se tendrá que implementar funcionalidades urgentes para el cliente, así se da comienzo al sprint. Esta metodología de trabajo no es fija, sino que al contrario, es dinámica pudiendo modificar la Product Backlog List en el desarrollo del sprint cuando se obtenga más conocimiento del software y con las retroalimentaciones del cliente, siendo el más completo y competitivo posible.

"Al comienzo de cada Sprint se hace la Reunión de Planificación del Sprint. Se divide en dos reuniones". (Deemer, Benefield, Larman, & Bas, 2009).

• Parte uno: ¿Qué vamos a hacer?

"La meta de la primera parte de la reunión de planeación del Sprint es adquirir los compromisos frente a las listas de requerimientos priorizadas estimando los esfuerzos." El dueño del producto elabora la lista de requerimientos que él quiere para ese sprint, todos los miembros conversan para tener claro lo que cada requerimiento supone en el sistema y despejar las dudas que pudiese haber. (Cifuentes Lozano, 2014). El cliente debe de indicar que espera del producto y de las entregas programadas, indicando los requerimientos funcionales y no funcionales, las tareas y los posibles bugs. Todos los miembros del equipo, incluidos el ScrumMaster y el propietario del producto, participan en reuniones de planificación de sprint.

Nadie más que el cliente será capaz de iniciar la planificación, aunque cualquier miembro del equipo puede agregar ítems. Estos requisitos del sistema son organizados en historias de usuario que luego serán descompuestas en niveles más pequeños llamados tareas, jerarquizadas de acuerdo al valor que dará al cliente.

Estas por recomendación de la experiencia de otros equipos deben de ser escritas y registradas en formatos que contengan su descripción, importancia, prioridad y valor. Tenemos un anexo A, para ver la plantilla de historia de usuario a usarse en esta propuesta de metodología híbrida.

Antes de iniciar cada sprint, se lleva a cabo una reunión de planificación para determinar qué características se incluirán en el sprint, que es priorizada por el propietario del producto. La primera vez que se produce esta reunión en un proyecto, se crea la acumulación de productos. Puedes pensar en esto como sprint 0. Las historias de usuario elegidas por el propietario del producto para ser incluidas en el sprint se dan al equipo y a través de una herramienta llamada Planning Poker, se redimensionan para mostrar la complejidad de una historia relacionada con las otras historias en grupo.

Planning Poker

"Técnica ágil de estimación y planificación basada en el consenso. Para comenzar una planificación de póker sesión, el propietario del producto o el cliente lee una historia de usuario ágil o describe una característica a los estimadores." Cada miembro del equipo cuenta con una seria de cartas con una valor, que representan el tiempo o dificultad en las que pueden ser implementadas, se empieza por exponer cada historia de usuario en detalle, si hubiese dudas se consulta con el dueño del producto, se procede a que cada miembro vote con una carta que ellos mismos seleccionen y todas son mostradas al mismo tiempo, si se ve que hay un consenso en la votación entonces, esa historia queda ya estimada, si en el caso de que hubiesen votaciones desiguales, se procede con la discusión de los valores por parte de cada miembro.

Quien estimo con el valor alto y quien estimo con el valor más bajo, discuten sus razones con los asistentes, entonces se procede nuevamente con la votación al mismo tiempo.

Este proceso se irá repitiendo hasta llegar al consenso, o se decide posponer la votación de esa historia hasta obtener mejor información. (Cohn, 2006).

La estimación grupal de las historias de los usuarios es una parte importante de la programación extrema (XP), utilizada tanto para la planificación de lanzamientos como para las iteraciones. La investigación ha demostrado que aunque la estimación grupal en muchos casos es superior a la estimación individual, todavía hay margen de mejora. Por ejemplo, el rendimiento de la estimación grupal se puede reducir mediante personalidades dominantes y efectos de anclaje. A través del análisis de 101 estimaciones de historias de usuarios, realizadas por un equipo de XP para la planificación de lanzamientos, investigamos si la introducción del proceso de estimación de póker de planificación mejoró la capacidad de estimación del equipo. Los resultados muestran que la planificación del póker mejoró el rendimiento de la estimación del equipo en la mayoría de los casos, pero que aumentó el error de estimación en los casos extremos. (Haugen, 2006).

Se establecen los roles:

Product Owner

Persona que comprende todo el funcionamiento del negocio y del equipo de desarrollo, por lo que debe de dar soporte a los miembros para asegurar un funcionamiento exitoso y también tener el conocimiento de los requerimientos del cliente para integrarlos y así dar valor al negocio.

Scrum Master

Persona encargada de brindar ayuda y soporte al equipo de desarrollo en el diario de sus actividades.

Scrum Team

Con la finalidad de trabajar en pares se recomienda que este grupo de integrantes sea en múltiplo de 2.

Encargado de Pruebas

Persona o grupo de personas que al estar notificadas de los problemas y errores surgidos en el desarrollo del sprint, empieza a testear a través de pruebas para encontrar soluciones y de ser posible documentarlas para el futuro uso.

SprintBacklog

Constituido por todos los requisitos del sistema que fueron manifestados por el cliente en las reuniones de planificación del sprint, en donde está el presente para la resolución de dudas en la elaboración y estimación de historias de usuario, para el nuevo sprint.

2. Metáfora del sistema

Cuando se comunica con la empresa, es útil tener un lenguaje común entre programadores y usuarios, para que los sistemas complejos se puedan explicar fácilmente. Nombres de métodos, clases y de variables extensos o muy cortos, confundirán al equipo de desarrollo y más aún al cliente, se recomienda el uso de nombres sugestivos y un glosario de términos para el correcto comprender de todo aquello que será usado.

Parte dos ¿Cómo lo Vamos a hacer?

"Los equipos comienzan a descomponer la lista de requerimientos seleccionados en tareas, que para entregarlas deben estar completas. Las tareas pueden ser: conseguir una entrada adicional del usuario, diseñar una nueva pantalla, adicionar nuevas columnas a la base de datos, etc." Se crea el Sprint Backlog que son todas las historias de usuario que integran la iteración, que tienen las descripciones y los requisitos necesarios.(Cifuentes Lozano, 2014).

Una vez que las historias de usuario son dimensionadas, se convierten en una serie de tareas por el equipo y una estimación de tiempo sobre cuánto tiempo tomará cada tarea. Una vez hecho todo esto, el equipo examinará la lista completa de trabajos enviados para el sprint y decida si pueden comprometerse a completar el trabajo al final del sprint. Para decidir esto, el equipo vota con cinco dedos para evaluar las opiniones de los miembros individuales.

Un miembro del equipo simplemente levanta la mano y, a través de la cantidad de dedos que sostiene, muestra lo que mejor refleja su nivel de confianza. Un valor manual de "1" significa que el miembro del equipo está Muy dudoso de la propuesta. Un valor manual de "5" significa que el miembro del equipo tiene mucha confianza en la propuesta. Si nadie tiene un valor de "1" o "2", entonces el equipo se compromete a ese trabajo para el sprint. Si se muestra un valor de "1" o "2", entonces el equipo discute por qué ese miembro del equipo votó de esta manera y ajusta la propuesta en consecuencia.

Una vez que el equipo se compromete a entregar la lista de historias de usuarios y tareas dentro del sprint, el ScrumMaster promulga una congelación de cambios para permitir que el equipo desarrolle la historia del usuario como se escribió anteriormente. Un backlog de Sprint está compuesto por todas las historias de usuarios y tareas requeridas para completar el sprint. Se llevan a cabo las siguientes actividades:

- **Detallar las Tareas:** Descomposición del Product Backlog List que va durar entre 4-16 horas.
- **Determinar responsabilidades:** las tareas y actividades deben de tener un responsable y cada miembro una responsabilidad.
- **Identificar los riesgos:** Anticipar requerimientos que pueden significar riesgos en el momento de la implementación.
- **Estimación del tiempo de cada miembro del equipo:** Los integrantes del equipo no solo se dedican al desarrollo de código sino que también tienen otras actividades dentro de la empresa o producto de sus necesidades, por eso es necesario calcular el tiempo en horas de cada miembro del equipo que servirá para programas tiempos de entrega.

- **Tablero de tareas o Task Board:** Constituido por 3 columnas, el antes, durante y después (por hacer, haciendo y hecho) que indican el estado de desarrollo que tiene una historia de usuario pero únicamente dentro de un sprint. Políticas explícitas así cada miembro conoce en dónde va el sprint, cada día actualizar la tabla.

- **Figura de Producto o Burn Up Chart:** Es importante esta herramienta para el dueño del producto y desarrolladores, para identificar en qué situación se encuentra el producto, brinda una cadena evolutiva de cada sprint, y claramente va a depender de la velocidad que el equipo tenga, el dueño del producto visualiza cómo va la evolución de sus requisitos y al estar ligado al desarrollo, provee información in situ frente al surgimiento de algún error.

  "Permite conocer al Product Owner las versiones previstas, las funcionalidades de cada una, velocidad estimada, fechas previstas, margen error y velocidad real." (Palacio, 2007).

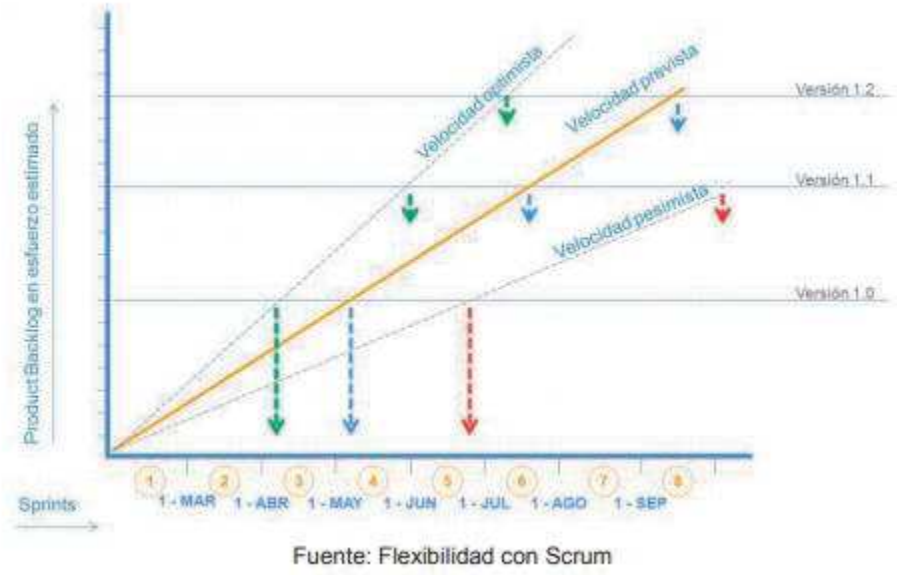


Figura 26. Burn Up Chart. (Palacio, 2007)

Tarjetas CRC (Clase Responsabilidad Colaboración)

Se utilizan las tarjetas CRC para representar las clases dentro de un proyecto, como parte de la descomposición de historias de usuario en tareas, componiéndose de un título en la parte superior, en la izquierda las responsabilidades y en la derecha las clases o métodos colaboradores con esta clase, se muestra en el Anexo B una plantilla de tarjeta CRC.

"El formato físico de las tarjetas CRC facilita la interacción entre los stakeholders, en sesiones en las que se aplican técnicas de grupo como tormenta de ideas o juego de roles, y se ejecutan escenarios a partir de especificación de requisitos, historias de usuario o casos de uso." Entonces con el transcurso del desarrollo de la metodología se van creando responsabilidades y colaboraciones de clases, estas tarjetas CRC adquieren métodos y atributos y una serie de características, que son escritas en las tarjetas. (Casas & Reinaga, 2009).

Valor: Responsabilidad

Distribuir responsabilidades a cada miembro del equipo, que los impulsará a completar objetivos y trabajos con calidad respecto a su responsabilidad.

3. Diseño simple

El diseño del sistema de alto nivel ocurre al inicio y durante una iteración. Una vez que finaliza la reunión de planificación, el equipo se reunirá sin el dueño del producto para discutir el diseño de alto nivel del sistema.

Usaremos las Tareas de ingeniería, al descomponer las historias de usuario en tareas que las hacen los programadores o el equipo de desarrollo, quienes ya entienden los términos técnicos de los mismos, se procede a utilizar la plantilla de tarjeta usada en el Anexo C, cada integrante puede ver que tareas hace y tener un control de ellas con el fin de luego retroalimentar el avance de las mismas en las reuniones.

"Un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione. Se sugiere nunca adelantar la implementación de funcionalidades que no correspondan a la iteración en la que se esté trabajando." (Joskowicz, 2008).

Valor: Sencillez – Simplicidad

Tareas sencillas. Pero en el transcurso del desarrollo del proyecto hay tareas que no pueden ser ejecutadas en esos términos, por lo tanto fueron divididas en iteraciones hasta que se reduzca su nivel de complejidad.

Tablero Kanban

Es una gran herramienta de ayuda que sirve para visualizar el proyecto de manera general, como parte de las políticas explícitas y el valor de la comunicación, se tiene un tablero con todas las historias de usuario, tareas y actividades. Tener una visión clara, sin confusiones, sin nada turbio o a medias tintas mejora la organización de una empresa, empodera a sus integrantes, los motiva para completar el flujo de trabajo con calidad.

Al trabajador con una sola metodología por ejemplo Scrum, estos tableros se reinician con cada sprint, quedan en cero a medida que vamos avanzado con el sprint, pero al utilizar este modelo híbrido propuesto, vamos a tener un flujo de entrada y salida de historias de usuario, tareas y actividades de manera continua

durante todo el proyecto, porque estas siguen el flujo de vida del proyecto: por hacer, haciendo y hechas, terminando en poder archivarlas para su posterior revisión.

Cartas Kanban

Se usó post it el cual ayudó en la identificación de las tareas que estaban por hacer, las que se están haciendo y las que ya se hicieron.

Valor: Visualizar flujo de trabajo

Al tener el flujo de vida del proyecto a la vista, poder consultarlo cuántas veces quieran, contribuye en el equipo a poder motivarlos, completar tareas de mejor manera y tener un producto de calidad.

4. Cliente en el sitio

Como punto fuerte de la metodología se tiene la disponibilidad del cliente 24/7 para poder consultar dudas sobre requerimientos, ya sea de manera física o manera virtual, al no tenerlos disponibles va a generar un mayor retroceso en el proyecto, mayor tiempo de solución de errores, producto en base al criterio de los programadores y no del cliente. El producto debe de cumplir con las expectativas del cliente.

Para el éxito del proyecto se busca la colaboración del cliente en grados altos, más que cumplir un papel, gracias a la flexibilidad y dinamismo ayuda al cumplimiento de un proyecto, más que seguir al pie de la letra guías ya impuestas antes de empezar. “Por ello, se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.” (Letelier & Penadés, 2006).

Valor: Enfoque al cliente

El proyecto tiene mayor fluidez y la satisfacción del cliente cuando Se cumplen con las expectativas del mismo. No se desarrolla software de acuerdo a los criterios del programador así tenga la experiencia necesaria, porque el producto será usado en primer lugar por los clientes quienes lanzarán el producto a

disposición de sus usuarios y un producto mal hecho, sin enfoque al cliente, significará desperdicio de recursos y tiempos.

5. Equipo sentados juntos

Tener al equipo centrado y con compromiso es primordial, debiendo estar dentro de la distancia en donde podrían "comunicarse" de gritos el uno al otro. Esto mejora la comunicación e imparte un sentimiento de camaradería.

"Entorno físico debe ser un ambiente que permita la comunicación y colaboración entre todos los miembros del equipo durante todo el tiempo, cualquier resistencia del cliente o del equipo de desarrollo hacia las prácticas y principios puede llevar el proceso al fracaso (el clima de trabajo, la colaboración y la relación contractual son claves), el uso de tecnologías que no tengan un ciclo rápido de realimentación o que no soporten fácilmente el cambio, etc." (Letelier & Penadés, Repositorio institucional de la Universidad de Las Tunas, 2012).

Sentarse juntos es esencial para una programación de pareja efectiva, quizás en un nivel más fundamental, es esencial reunir al equipo junto con los clientes para una comunicación fácil y efectiva, sentarse juntos aumenta la comprensión compartida del equipo y ayuda a construir el espíritu de equipo, que es crucial al comprometerse como equipo para un proyecto. Crear espíritu de equipo es importante porque, con el tiempo, las relaciones comienzan a formarse con los miembros del equipo. Las buenas relaciones y el espíritu de equipo realmente aumentan la productividad. Todas las tareas son manejadas por el equipo en lugar de "arrojado por la pared" a alguien que no conoces, una gran parte del vocabulario personal está en el lenguaje corporal, y esto solo se puede notar al sentarse juntos.

Valor: Comunicación

Aplicado desde la elaboración de los requerimientos e historias de usuario, retroalimentado en las reuniones diarias, de planificación y de retrospectiva. Contribuye el tener a todos los miembros del equipo junto con los clientes en un espacio físico común con el fin de maximizar la comunicación, porque aunque un cliente no hable, el equipo debe notar los gestos que pueda expresar.

BurnDown Chart

Como objetivo este gráfico que será presentado al cliente, tiene la función de ver qué velocidad o cuáles fueron las razones por la que una funcionalidad no se terminó a tiempo.

El Scrum Master debe estar en permanente contacto con los miembros del equipo, para poder acelerar la solución de problemas o errores, supervisando a los integrantes en el cumplimiento de sus actividades, para tener una visión transparente del proyecto.

Scrum Diario

"Es una reunión que se realiza aproximadamente por un periodo de 15 minutos diariamente con el objetivo de hacer un resumen de las actualizaciones, y coordinaciones entre los miembros del equipo." Se expone básicamente, qué se hizo ayer, que se hará hoy y qué problemas han surgido, así todos los miembros tienen una idea del proyecto y saben hacia dónde ir, con el fin de cumplir y desarrollar el software.

Se supervisa la actualización de los gráficos y del tablero Kanban, para su respectivo análisis, usando las Cartas Kanban.

De preferencias las reuniones deben ser organizadas en el mismo ambiente donde se tiene el tablero Kanban, que facilita la explicación y visualización de las cartas Kanban, se sugiere mantener una buena organización y actualización del gráfico y tablero de avance.

6. Programación en pareja

El invertir en personal para realizar una programación en parejas, disminuye costos y errores a mediano plazo, mitigando errores y obteniendo calidad de código, mientras trabajen juntos la velocidad de trabajo se duplica, la organización y toma de decisiones se mejora. Con programación en pares, el código y las revisiones se convierten en "tiempo real", produciendo un nivel de calidad que deja de ser "solo de inspección". Con dos personas en desarrollo, es muy poco probable que ambos pasar por alto el mismo error.

Al integrar nuevos miembros al equipo, aún después de haber comenzado las iteraciones, el trabajar en pares ayuda a su rápida integración, aprendizaje y colaboración.

"Otra ventaja adicional en el mundo de la empresa de tener dos programadores realizando tareas conjuntamente es por la alta movilidad laboral. Es bien sabido, que muchos proyectos pierden un tiempo valiosísimo cuando uno de los integrantes del equipo deja el trabajo." La programación en pares busca la continuidad del desarrollo del proyecto así uno de ellos, ya no trabaje, falte o se encuentre de vacaciones, porque cada miembro de la pareja tiene el conocimiento necesario para continuar con el trabajo del otro, para cumplir con la entrega de funcionalidades previstas. (Robles & Ferrer, 2002).

7. Limitar el WIP

Restringir la cantidad de actividades, tareas e historias de usuario en las columnas del tablero, ayuda en gran medida a culminar y cumplir con los tiempos, motivo a los equipos, los preparo para cumplir con tareas que exigían enfoque y concentración, ayudó al equipo en continuar con el flujo de trabajo, sin estrés en la medida de observar tablero con decenas de cartas.

"Debemos tener en cuenta que limitar el WIP no se aplica solo a la fase de desarrollo, sino que también las fases previas de definición y análisis de tareas deben tener un cierto límite WIP." En la práctica, del desarrollo de software se adquiere conocimiento para tomar mejores decisiones en las iteraciones siguientes. Por lo que no es aconsejable limitar el Wip a muy pocas actividades al principio, se debe de tener un cierto número de historias de usuario a realizar para que el equipo se siente cómodo, por eso es mejor comenzar con un numero un poco alto para luego reevaluar en las siguientes iteraciones. (Bermejo, 2011).

8. Propiedad del código colectivo

A través de la programación en pares, el equipo produce código sin ser ellos los dueños de este, el código es del equipo y cada miembro puede arreglar y trabajar en cualquier parte de la base del código.

Gracias al valor que Xp brinda todos los miembros generan código y no hay la necesidad de hacer propiedad del producto obtenido, al rotara los miembros a distintas área del proyecto, todos conocen de todo y todos puede hacer los cambios que crean convenientes, para poder obtener un producto de valor.

Por esta razón si un miembro presenta alguna dificultad, el hecho de haber rotado por esa área le brinda la capacidad y experiencia para poder ayudar y solucionar juntos el error.

La propiedad del código compartida promueve que todo el personal pueda involucrarse para corregir y ampliar cualquier parte del código, además, todo lo anterior en conjunto con las pruebas frecuentes garantizan que los errores sean detectados de manera temprana. (Balza, Briceño, Lobo, & Nuñez, 2017).

9. Normas de codificación

Aplicar estándares de desarrollo de código, con simplicidad ayuda en tener código funcional claro y coherente.

Estas normas ayudan a los miembros del equipo, cuando les toque rotar y trabajar en otra funcionalidad y así un miembro podrá continuar con el desarrollo.

Al utilizar identificadores de variables nemotécnicos se facilita en gran medida la continuidad del desarrollo de software, pues al observar cada variable tendremos una idea del uso que esta tiene en el sistema.

“Para que estas cosas no pasen y tu código siempre esté a la altura, debemos seguir siempre un estándar de codificación o Code Standard en inglés. Los estándares de código, son parte de las llamadas buenas practicas o mejores prácticas”, siendo un conjunto de normas y medidas que van siendo perfeccionadas por los desarrolladores a través de la práctica, garantizando el

incremento de calidad de código, siendo estándares de escritura, formatos y denominaciones de código que dependen del lenguaje de programación a utilizar. (ohmyroot!, 2017).

Tenemos así:

- strDireccion (str = variable de tipo string + Nombre Descriptivo de la variable)
- intDNI (int = variable de tipo entero + Nombre Descriptivo de la variable)

10. Pruebas

Buscando entregar calidad de producto, Xp nos ayuda aquí con sus pruebas. Las pruebas comienzan identificando las características mínimas que debe de tener una funcionalidad para que sea aceptada, que serán la base de la construcción de pruebas unitarias e iniciar en estilo Desarrollo basado en Pruebas. Es preferible automatizar todas estas pruebas para posteriores usos dentro y en nuevos proyectos.

Existe una prueba llamada del semáforo, que usa los colores de este, si está en rojo, la funcionalidad está mal hecha, por otro lado, si la prueba falla y se desarrolla el código mínimo para pasarla es ámbar, en ambos casos el código se re factoriza y si la prueba pasa entonces está en verde pero se continua con la re factorización.

Al terminar al terminar la funcionalidad, existe un equipo de testeo para verificar que pasen por las respectivas pruebas, pudiendo estas personas pertenecer al proyecto o no, si encontrasen errores o mal funcionamiento, deben de ser reportados para que los programadores trabajen en solucionar y obtener un código más coherente. y actualizando en el tablero si existiesen errores o mal funcionamiento, estos son detenidos para poder ser consultados en las reuniones diarias de la iteración, si aquí los clientes decidían que estos errores pueden esperar se los deja para la siguiente iteración de lo contrario se organizaba para resolverlas al día siguiente.

Re factoring

Tenemos un código mucho más limpio y optimizado removiéndose código innecesario, comentado para otros programadores.

Las refactorizaciones son ampliamente reconocidas como formas de mejorar la estructura interna del software orientado a objetos mientras se mantiene su comportamiento externo. (Du Bois, Demeyer, & Verelst, 2004).

Una buena práctica para el re factoring es: La primera vez que hacemos algo, lo hacemos, la segunda vez igual lo haremos pero notaremos que estamos duplicando código y la tercera vez que hagamos lo mismo, se re factoriza.

11. Integración continua

Para garantizar que el código realmente funcione, el equipo se reúne a menudo y temprano. Un servidor de integración continua (CI) extraerá la base de código de un control de origen, lo compilará y ejecutará todas las pruebas automatizadas para garantizar que la compilación este correcta. La salida del servidor CI se puede notificar instantáneamente a desarrolladores, clientes y gerentes de proyecto. Este informe incluye: El número de pruebas aprobadas / reprobadas en este punto. El seguimiento diario proporciona un Indicador de progreso en la creación de valor para el cliente. Este bucle de retroalimentación rápida también permite a los desarrolladores arreglar la compilación. Cuanto antes se identifique un problema, es más barato arreglarlo.

"El proceso de integración de componentes que se requiere en los proyectos no es una tarea simple. La integración de software es un problema complejo sobre todo en sistemas que involucran código desarrollado por diferentes personas", es así que la integración continua proporciona resultados para el sistema, con los cuales podemos ver la función de cada módulo individual y colectivamente, Xp garantiza el uso de integración continua para tener código sin errores y coherente. (Salamon, y otros, 2014).

Se hace como parte del mantenimiento del código aún en etapa de desarrollo, se integra código correcto al sistema, a través de módulos o funcionalidades, ir de a

pocos integrando código re factorizado y aprobado por pruebas, con el fin de tener menos errores en la integración, integrar el sistema más rápidamente, garantizando supervisión constante para un producto de calidad.

Incrementos del producto

Cumplido el tiempo fijado y establecido, el equipo de desarrollo estará en la capacidad de entregar una funcionalidad o un incremento para el sistema que podrá ser utilizado por el cliente, siempre y cuando cumpla con las expectativas del mismo, cada finalizado un sprint se habrán cumplido con tareas, funcionalidades e historias de usuario.

En este marco de trabajo híbrido se debe de entregar funcionalidad para que el cliente ya pueda utilizar, es decir que ya se encuentra en el tablero, esta tarea, funcionalidad o historia de usuario en la columna de "Hecho".

El corazón de Scrum es un Sprint, es un intervalo prefijado durante el cual se crea un incremento de producto "Hecho o Terminado" utilizable, potencialmente entregable.

12. Ritmo sostenible

Debido a los cortos ciclos de liberación y la naturaleza iterativa de XP, recopilación de requisitos, diseño, desarrollo, pruebas e implementación de todo, sucede a menudo. Quiere decir que el trabajo debe de tener un ritmo, sin apresurarse ni retrasarse, por eso los errores o mal funcionamiento encontrados que no sean críticos podrán ser puestos en cola para la siguiente iteración. Por lo tanto, ahorran mucho tiempo si se compara con otra metodología ágil que tienen claramente tiempos definidos para cada etapa, dejando comentarios y pruebas para el final. El ritmo sostenible ayuda a la gerencia a planificar más eficientemente las vacaciones de todo el personal, faltas no planificadas y ocasionales "incendios" de producción que necesita ser extinguido inmediatamente.

"La metodología XP indica que debe llevarse un ritmo sostenido de trabajo. Anteriormente, ésta práctica se denominaba Semana de 40 horas. Sin embargo, lo importante no es si se trabajan, 35, 40 o 42 horas por semana." Tener un constante

y estable ritmo de trabajo hará que el equipo pueda desarrollar código de calidad sin tener que apresurar tiempos y gastar más recursos de los necesarios. (Rosas, 2017).

Sprint

Procedimiento de adaptación de las cambiantes variables del entorno (requerimientos, tiempo, recursos, conocimiento, tecnología). Son ciclos iterativos en los cuales se desarrolla o mejora una funcionalidad para producir nuevos incrementos. Durante un Sprint el producto es diseñado, codificado y probado."Los objetivos a cumplir dentro de cada sprint deben ser redactados de la manera más simple y corta posible para que todos lo comprendan y lo recuerden.

Las actividades que se desarrollan durante del Sprint son: Sprint Planning Meeting, Sprint Backlog, Daily Scrum Meetings y Sprint Review Meeting.

Construir las historias de usuario para el equipo de trabajo, gracias a los requerimientos expresados por el cliente, para empezar a dar al cliente, funcionalidades a través de entregables continuos. Teniendo un ritmo sostenible, realización de pruebas e integración continua al sistema, son los objetivos que se tiene un cada sprint con la finalidad de entregar un proyecto final, el más completo y de calidad posible para el cliente.

Como parte del sprint se tiene un conjunto de historias de usuario que han sido estimadas y valoradas sin exceder un WIP, para que al final de la iteración se pueda entregar funcionalidad utilizable para el cliente, así garantizar compromiso y responsabilidad.

El equipo de desarrollo empieza con la programación de las tareas. Con esta propuesta de modelo de gestión de trabajo, estos equipos tienen características de auto organización y toma de decisiones si algún requerimiento genera dudas o problemas.

Durante el desarrollo de las tareas deben pasar por las pruebas unitarias de software, se desarrolla sólo el código mínimo para que la historia de usuario pase las pruebas, los pasos son:

- Utilizar una historia de usuario, detallada, estimada y valorada.
- Empezar a elaborar el código para las pruebas.
- Elaborar código para que la funcionalidad pase la prueba.
- El equipo del test aprueba o desaprueba el éxito del caso de prueba, surgiendo nuevos casos de pruebas y afinamientos.
- Re factorización del código para ir mejorando el sistema paulatinamente.

Por tanto la funcionalidad al fin del sprint cumple con las expectativas del cliente, es recomendable hacer la integración modular o funcional, sin acumular para facilitar el tiempo de integración y lanzamiento de software funcional por cada sprint terminado, con criterios de re factorización, corrección y mejora de código para las pruebas de aceptación del cliente.

Release Planning Meeting (Reunión de Entrega)

"Se establece el plan y los objetivos de la entrega, además de decir la estrategia que el equipo Scrum y el resto de la organización van a seguir." Se busca la respuesta a las preguntas:

• ¿Cuál será la mejor manera para obtener un producto de calidad?

• ¿Cómo cumplir con las expectativas del cliente sin tener que exceder el gasto de recursos?

Se acuerdan los objetivos a cumplir, los tiempos y las historias que serán entregadas, qué estimación es la adecuada para el equipo, aunque no es una reunión obligatoria pero ayuda en la comprensión de obligaciones con el cliente.

Re Release Planning Meeting

Reunión con el cliente para revisar y re plantear las prioridades del proyecto.

El problema de planificación de lanzamiento (RP) puede investigarse desde dos dimensiones: qué liberar y cuándo liberar. Investigamos la decisión de "qué" publicar en términos de qué nuevas características o solicitudes de cambio deben asignarse e implementarse en qué versiones de un sistema de software. El RP para sistemas en evolución es un desafío, porque las nuevas características pueden

requerir cambios en el sistema existente. Una desventaja importante de los métodos de RP existentes es que no consideran los sistemas existentes al tomar decisiones de RP. (Saliu & Ruhe, 2007).

Planificación de la siguiente iteración

Al finalizar la producción del sprint, se convoca a una reunión en donde se hará la planificación para la siguiente iteración, con todos los involucrados en el desarrollo del producto, para analizar experiencias, observaciones, dudas.

Sprint Review Meeting

Realizada al final del sprint, se analiza las historias de usuario seleccionadas para el siguiente sprint, si cabe la posibilidad de que alguna necesite mejora o cambios, es prudente hacerlos antes del inicio del sprint, es una excelente reunión para hacer retroalimentación de la evolución del proyecto. La razón de ser de esta reunión es que mediante la retroalimentación se genera un ambiente de confianza y transparencia tanto para los miembros del equipo como para el cliente.

Para Sprints más cortos, esta reunión tenderá a ser más corta. Este evento es organizado por el Product Owner, y es necesaria la presencia de todo el equipo de Scrum. El rol del Scrum Master es asegurar que el evento ocurre y que cumple los tiempos establecidos.

Como función importante del dueño del producto es ir explicando que historias de usuario del Product Backlog ya han terminado y cuáles aún faltar por empezar y terminar. Se debe de ir actualizando el Tablero.

Consta de:

- Presentar la funcionalidad terminada y lista para usar.
- Ver el por qué no se terminaron o cumplieron las tareas y actividades.
- Ver en la medida si se encuentra problemas, para generar un cambio.

Retrospectiva

Esta reunión se da después de acabar una iteración y luego de hacer la reunión de revisión del sprint. Esta reunión tiene como característica el de mirar hacia atrás, prima la autocrítica por parte del equipo de trabajo, se ve qué cosas se hizo bien para continuar haciéndolas, que cosas se hizo mal para corregir en el siguiente sprint, se plantea puntos de mejora.

Como se trata de un enfoque híbrido, se va a generar código con calidad, porque aprovechará las mejores prácticas de las tres metodologías, facilitando así el trabajo del equipo de desarrollo.

Aunque es frecuente realizarlas al final de cada sprint, no deben confundirse con las reuniones de revisión del sprint.

Como es una reunión de autocrítica y de ver hacia atrás, lo que se ha realizado, debemos de analizar por qué ciertos procesos han salido bien y porque otros no, preferible documentar los errores para cuando se vuelva a presentar ya tenemos la solución más rápida, si se cumplió con las expectativas del cliente, si se obtuvo una funcionalidad de calidad con código re factorizado, sencillo, coherente. Obteniéndose un plan de acciones de mejora y un acuerdo de equipo actualizados.

Última ceremonia de un Sprint. Oportunidad para el equipo de inspeccionarse a sí mismo, y crear un plan de mejora para el siguiente Sprint. (Tesei, Cabrera, Vaquero, & Tedini, 2019).

13. Pequeños lanzamientos

Como una práctica de esta propuesta de metodología híbrida se promueve los lanzamientos continuos y constantes de funcionalidades hechas y listas para usar por el cliente, no importa siendo estos pequeños, sino que den valor al negocio, mientras más entregables sean, mucho mejor. XP no permite que el equipo de desarrollo se oculte durante meses, esperando que el proyecto se complete a tiempo. Esta transparencia de frecuentes lanzamientos alienta los dueños del producto, que se está cumpliendo con sus expectativas y dando valor a su negocio.

Así el cliente puede obtener ventajas competitivas al tener mejores y nuevas funcionalidades para sus usuarios.

“Por lo tanto, partiendo de todas las características aquí expuestas, se puede definir una metodología ágil, como aquella que busca y prioriza la satisfacción del cliente a través frecuentes y pequeños evaluables de software, facilitando de esta manera, una comunicación fluida entre todos los participantes” Sin importar el ron que cumplan todos deben de tener una comunicación con respeto y fluida, para garantizar un producto de calidad, los sprint garantizan entregables funcionales para el cliente, es así que se obtiene mejora continua y constante durante la elaboración del proyecto. (Moré Martín, 2011).

CAPÍTULO IV

ANÁLISIS Y RESULTADOS DE LA INVESTIGACIÓN

4.1. ANÁLISIS DE FACTIBILIDAD DE IMPLEMENTACIÓN DE LA METODOLOGÍA

Estudio de factibilidad para la aplicación de la metodología

Como parte de implementar una nueva herramienta en una institución que use poco o mucho las metodologías ágiles, es importante conocer sí sería factible o no la implantación de un conjunto integrado de prácticas como lo es esta propuesta de metodología híbrida, por lo cual realizaremos una encuesta en la institución para proceder a aplicar esta propuesta de marco de desarrollo ágil.

Para poder adaptar esta propuesta se ha llevado a cabo a través de un cuestionario.

Esta encuesta estará enfocada en diversos aspectos de una institución, cómo es que manejan al personal, si utilizan alguna metodología, o realizan pruebas a sus productos, nivel de jerarquía en su institución, teniendo el siguiente gráfico de puntajes:

RESPUESTA	PUNTAJE
Generalmente no se utiliza o se usa en situaciones concretas	< 40 puntos
Se usa en la mayor parte de situaciones	40 – 60 puntos
Usada en casi todas las áreas	60 – 85 puntos
Usada de forma total	> 85 puntos

Figura 27. *Valoración de tabla de puntajes de la encuesta*

Valoración menor de 40 puntos: Al no usar metodologías ágiles, se debe de corregir e implantar una metodología para mejorar la calidad de sus productos.

Si el puntaje esta entre 40-60: Aún hay falencias en el uso de su metodología.

Si el puntaje esta entre 61-85: Tiene muy pocos errores en el uso de la metodología.

Mayor a 86 puntos: Su institución tiene la capacidad y aptitud para poder implantar una metodología híbrida ya que todos sus miembros aplican correctamente las prácticas.

Como análisis de los resultados obtenidos, se procede a la recomendación de implantar esta propuesta de metodología híbrida a aquellas instituciones que han obtenido un puntaje mayor a los 70 puntos. En el Anexo D se encuentra el modelo de la encuesta.

4.2. CONFORMACIÓN DEL EQUIPO

Se cambian los miembros y conformación de los equipos ya que en las metodologías tradicionales, tienen estructuras jerárquicas muy excesivas y sin cambios, cada miembro hace lo que le toca y nada más.

Por lo tanto en esta propuesta de marco de trabajo la conformación de estos equipos, se rompe con lo tradicional, son equipos multidisciplinarios, con capacidad de toma de decisiones de parte de cada uno de los miembros, con políticas explícitas y código compartido, todos conocen el código porque trabajan en pares y tienen una rotación dentro del proyecto. Entonces se convierten en equipos ágiles, con empoderamiento y flexibilidad en la producción de software de calidad.

El equipo para esta propuesta de modelo de gestión de trabajo ágil será un equipo multidisciplinario, no sólo con actitudes para la programación sino también que contarán con conocimiento amplio para las otras actividades que el equipo de trabajo necesite, así como el de comunicación con sus compañeros y con el cliente, un equipo solidario con sus miembros que significará el mínimo tiempo en la solución de errores, código compartido para obtener código de calidad.

4.3. CAPACITACIÓN

Al tratarse de una propuesta de marco de trabajo se debe programar una serie de charlas y capacitaciones de los siguientes temas:

- Qué son las metodologías ágiles
- Qué es el Manifiesto Ágil
- Metodología XP, conceptos, aspectos, prácticas etc.
- Metodología Scrum, conceptos, aspectos, prácticas, etc.
- Resumen de Metodología Kanban
- Marco de trabajo ágil
- Roles
- Actividades
- Artefactos
- Valores
- Prácticas y Técnicas

4.4. DESARROLLO DE LA METODOLOGÍA

4.4.1. JUEGO DE PLANIFICACIÓN

Como primer puntos se cronograma una reunión con el cliente, quien entregó al equipo de desarrollo la lista de requerimientos y funcionalidades del proyecto y priorizando los requerimientos para la iteración. Se resolvieron las dudas junto con el cliente, en aquellas funciones más importantes. Se hizo el compromiso de los integrantes del equipo con la realización del proyecto, se establecen los roles.

Se procede a construir el Product Backlog de la siguiente manera

Debe permitir registrar a todo el personal de la institución.	Debe mostrar el logo de la institución.	Debe permitir mover al personal modificar sus datos.
Debe poder registrar todo el equipo, inventario de la institución.	Debe poder cambiar el equipo, inventario, de la institución, en cuanto a sus características.	Debe permitir generar reportes de todo el inventario de la institución.
Debe permitir generar reportes de inventario por área para el control.	Debe permitir generar reportes de inventario por persona para el control.	

Se procedió a elaborar las historias de usuario:

a. Historia de usuario N°01: Agregar nuevo trabajador
b. Historia de usuario N°02: Cambiar datos del trabajador
c. Historia de usuario N°03: Agregar nuevo equipo
d. Historia de usuario N°04: Cambiar datos de equipo
e. Historia de usuario N°05: Listado de todos los equipos
f. Historia de usuario N°06: Listado de todos los equipos de un área
g. Historia de usuario N°07: Listado de todos los equipos de un trabajador de un área

HISTORIA DE USUARIO N° 03

HISTORIA DE USUARIO	
Número: 03	**Nombre de la historia:** Agregar nuevo equipo
Usuario: Trabajadores de la institución	**Prioridad:** Alta
Descripción: El encargado de la oficina de patrimonio quiere que se guarden todos el equipo y demás enseres existentes en la institución, así como los que serán recién adquiridos, todas sus características como son nombre, color, medida, marca, modelo, serie, estado, así como a qué oficina y trabajador pertenece.	
Observaciones: Los muebles no tienen serie, pero todo el inventario tendrá un registro único que se identificará por un código de barra, pegado en cada inventario.	

HISTORIA DE USUARIO N° 07

HISTORIA DE USUARIO	
Número: 03	**Nombre de la historia:** Listado de todos los equipos de un trabajador de un área
Usuario: Responsable de oficina de Patrimonio	**Prioridad:** Alta
Descripción: El responsable de la oficina de patrimonio quiere que se pueda obtener un listado de todo el inventario que pertenece a un trabajador X, qué trabajador lo registró, indicando todas sus características como son nombre, color, medida, marca, modelo, serie, estado, así como a qué oficina y trabajador pertenece.	
Observaciones: Este listado tendrá validez para rendir informe de inventario al área de patrimonio.	

Procederemos a usar la técnica del Planning Poker para la estimación de las historias de usuario, cuando todo el equipo está alineado, tengamos el material necesario para la sesión y entiende la tarea que se tiene que puntuar.

Cada miembro del equipo ya conoce las historias de usuario entonces se procede a estimar cada una. Si por alguna cuestión, un miembro del equipo no conoce las historias de usuario, se debe de iniciar esta reunión con una explicación y retroalimentación de las historias que se tendrán que estimar.

El moderador de la reunión comienza por leer cada historia de usuario, al punto de que cada miembro tenga claro qué es lo que se hará con esa historia de usuario, entonces cada miembro del equipo a excepción del Scrum master y el dueño del producto, tienen que estimar, para lo cual cada uno elegirá una carta en la medida del esfuerzo personal que crean en desarrollar esa historia de usuario.

Se genera una discusión si no existe acuerdo. Explicando el por qué eligieron esa carta y por qué creen que la historia de usuario tiene esa dificultad, tanto el que puntuó con menor carta y el que estimo con la carta más alta, se procede a estimar nuevamente, si aún no hay un acuerdo por parte de los miembros, la tercera vez se recomienda elegir el de la máxima estimación.

Terminando esta reunión ya se tiene las historias de usuario estimadas y validadas por cada miembro del equipo, para cada una de las siete historias de usuario.

HISTORIA DE USUARIO	**ESTIMACIÓN**
Historia de usuario N°01: Agregar nuevo trabajador	5
Historia de usuario N°02: Cambiar datos del trabajador	3
Historia de usuario N°03: Agregar nuevo equipo	8
Historia de usuario N°04: Cambiar	3

datos de equipo	
Historia de usuario N°05: Listado de todos los equipos	13
Historia de usuario N°06: Listado de todos los equipos de un área	13
Historia de usuario N°07: Listado de todos los equipos de un trabajador de un área	13

Se procede a la planificación de esta primera iteración, con ayuda del Scrum master, para garantizar el empleo de tiempos y menores esfuerzos, que se logró gracias al uso del diseño simple, elaborándose el SprintBacklog:

ITERACIÓN	HISTORIAS DE USUARIO	ESTIMACIÓN	TOTAL
1	Historia de usuario N°03: Agregar nuevo equipo	8	16
	Historia de usuario N°01: Agregar nuevo trabajador	5	
	Historia de usuario N°02: Cambiar datos del trabajador	3	
2	Historia de usuario N°05: Listado de todos los equipos	13	16

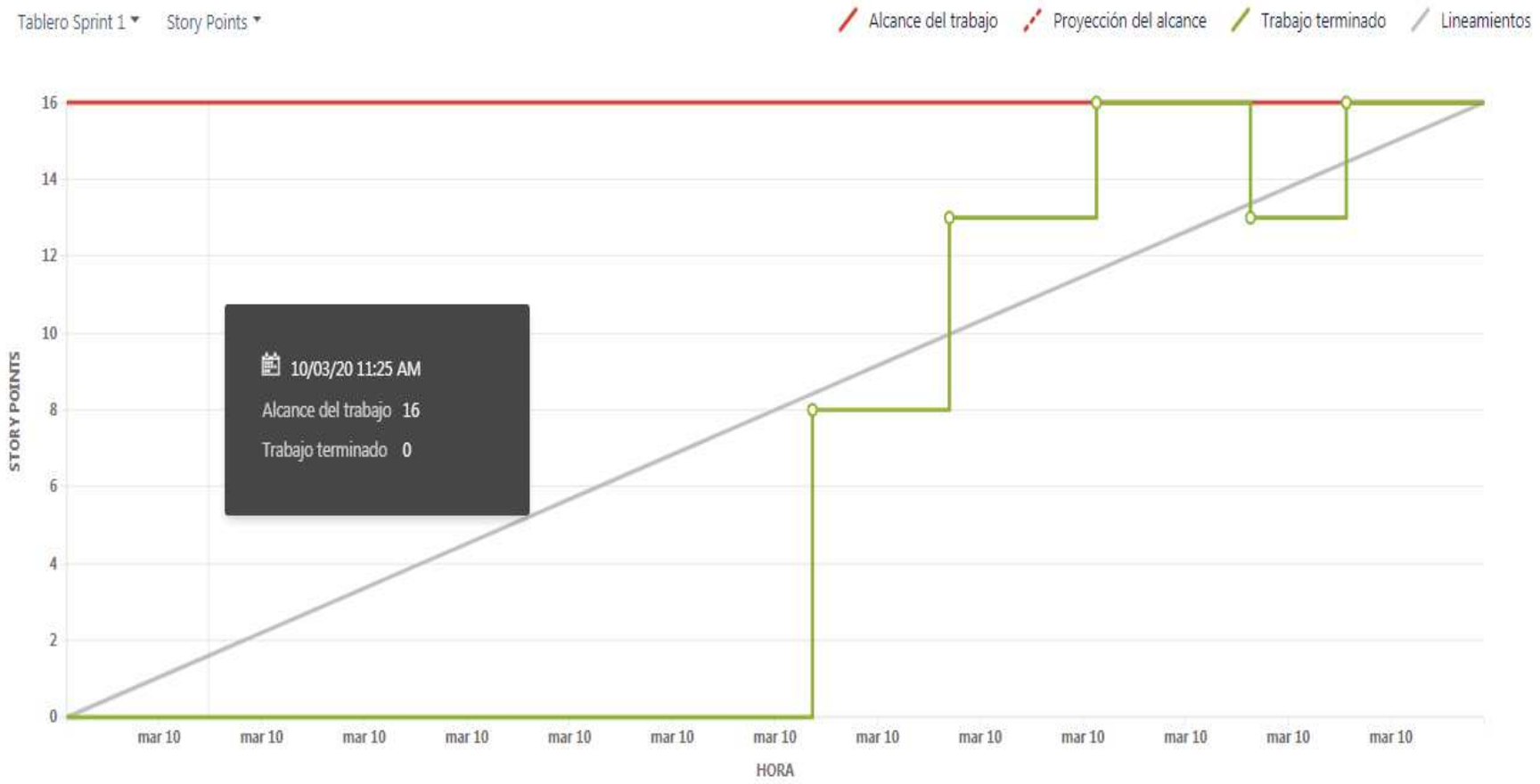
	Historia de usuario N°04: Cambiar datos de equipo	3	
3	Historia de usuario N°07: Listado de todos los equipos de un trabajador de un área Historia de usuario N°06: Listado de todos los equipos de un área	13 13	26

4.4.2. Metáfora del sistema

Luego de la descomposición de los requerimientos en tareas, determinando responsabilidades, riesgos, estimación de tiempo. Se procede a elaborar el Task Board para este Sprint.

HISTORIA DE USUARIO	PARA HACER	EN PROCESO	PARA PRUEBAS	HECHO
Historia de usuario N°03: Agregar nuevo equipo **Historia de usuario N°01: Agregar nuevo trabajador** **Historia de usuario N°02: Cambiar datos del trabajador**	Elaborar interfaz Cambiar datos del trabajador. Programación de botones.	Elaborar interfaz registro de inventario. Programación de los botones.	Elaborar interfaz registro de persona. Programación de los botones.	Elaborar esquema de la base de datos.
Historia de usuario N°05: Listado de todos los equipos **Historia de usuario N°04: Cambiar datos de equipo**				
Historia de usuario N°07: Listado de todos los equipos de un trabajador de un área **Historia de usuario N°06: Listado de todos los equipos de un área**				

Gráfico Burn up Chart



Los programadores descomponen las historias de usuario que fueron elegidas para la iteración, en tareas. Duró, esta planificación, 2 semanas, que para casos de horario laboral fueron 7 días, por lo que no se rompe el esquema de esta propuesta de modelo de gestión de trabajo de tener iteraciones cortas de trabajo.

TARJETAS CRC

NOMBRE DE LA CLASE	Guardar_Inventario
RESPONSABILIDADES	**COLABORADORES**
Permite crear, modificar, editar, eliminar los registros del inventario. Gestiona los datos registrados para su registro en la base de datos.	Guardar_Persona

Fuente: (Chiluisa & Loarte, 2014)**.**

4.4.3. Diseño simple

Se elabora las Tareas de Ingeniería con el fin de determinar las tareas que cada miembro del equipo tiene.

Tareas de Ingeniería

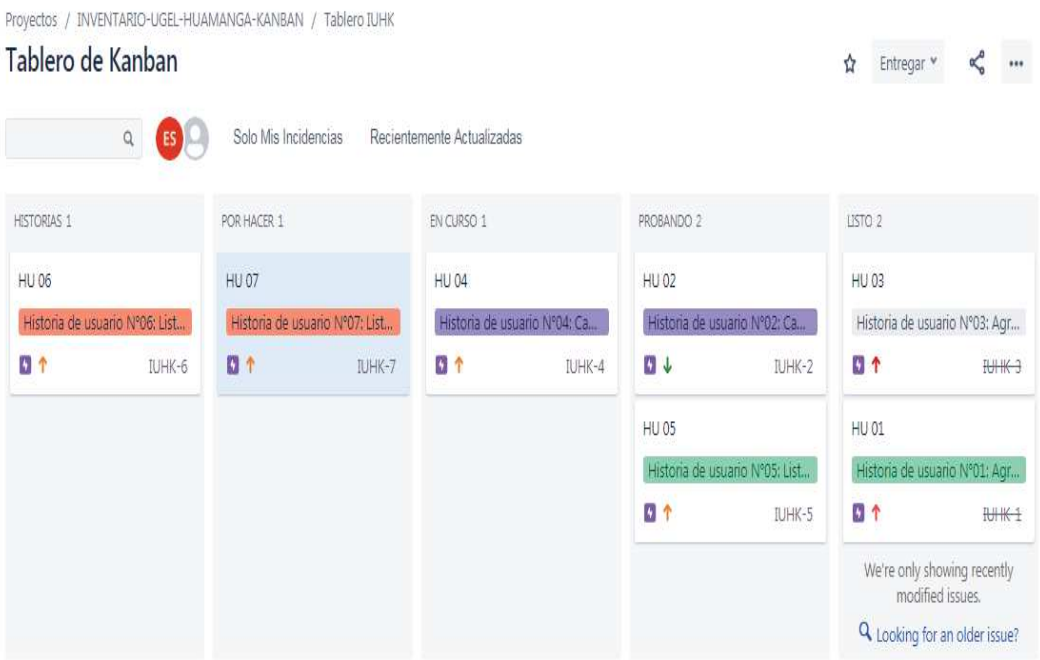
TAREA: Registrar_Inventario	
Nº DE TAREA: 01	**Nº DE HISTORIA:** 03
TIPO DE TAREA	Desarrollo
FECHA INICIO: 11/03/2019	**FECHA FIN:** 18/03/2019
PROGRAMADOR RESPONSABLE	Eder Solórzano Huallanca
DESCRIPCIÓN	
El sistema permitirá registrar el inventario que pertenezcan a la Ugel Huamanga. Los datos que se requiere para el registro de inventario son: Código (se manejara un código interno), nombre, en caso de que sea necesario llenar marca, modelo, serie, color, también indicar la cantidad, el estado (malo, regular, bueno, nuevo) y la fecha de registro. Si desea indicar una descripción. También puede añadir una foto del inventario.	

Dar clic en el botón guardar si todo está correcto, guardara el nuevo inventario, caso contrario aparece un mensaje de error. Dar clic al botón nuevo si se desea limpiar los campos y registrar un nuevo inventario. Dar clic al botón cancelar para salir de esta ventana.

Fuente: (Ferreira, 2013)**.**

"Mantenemos el diseño simple desde el principio y mejorándolo continuamente, en lugar de conseguir que todo funcione desde el principio y entonces congelarlo. Esto lo estamos haciendo bastante bien, es decir, empleamos una cantidad razonable de tiempo re factorizando y mejorando los diseños existentes, y rara vez empleamos tiempo haciendo grandes diseños desde el principio." Es parte del equipo, y misión del Scrum master el de orientas y guiar en las prácticas de la metodología con el fin de en caso encontrar problemas y errores, estos se solucionen de la manera más rápida posible para evitar retrasos y gastos innecesarios en el proyecto, satisfacer las expectativas del cliente es un objetivo del proyecto. (Henrik, 2007).

Con el fin de tener una correcta visualización del flujo del trabajo de todo el proyecto se procede a elaborar el Tablero Kanban.



Fuente: Elaboración propia.

4.4.4. Cliente en el sitio

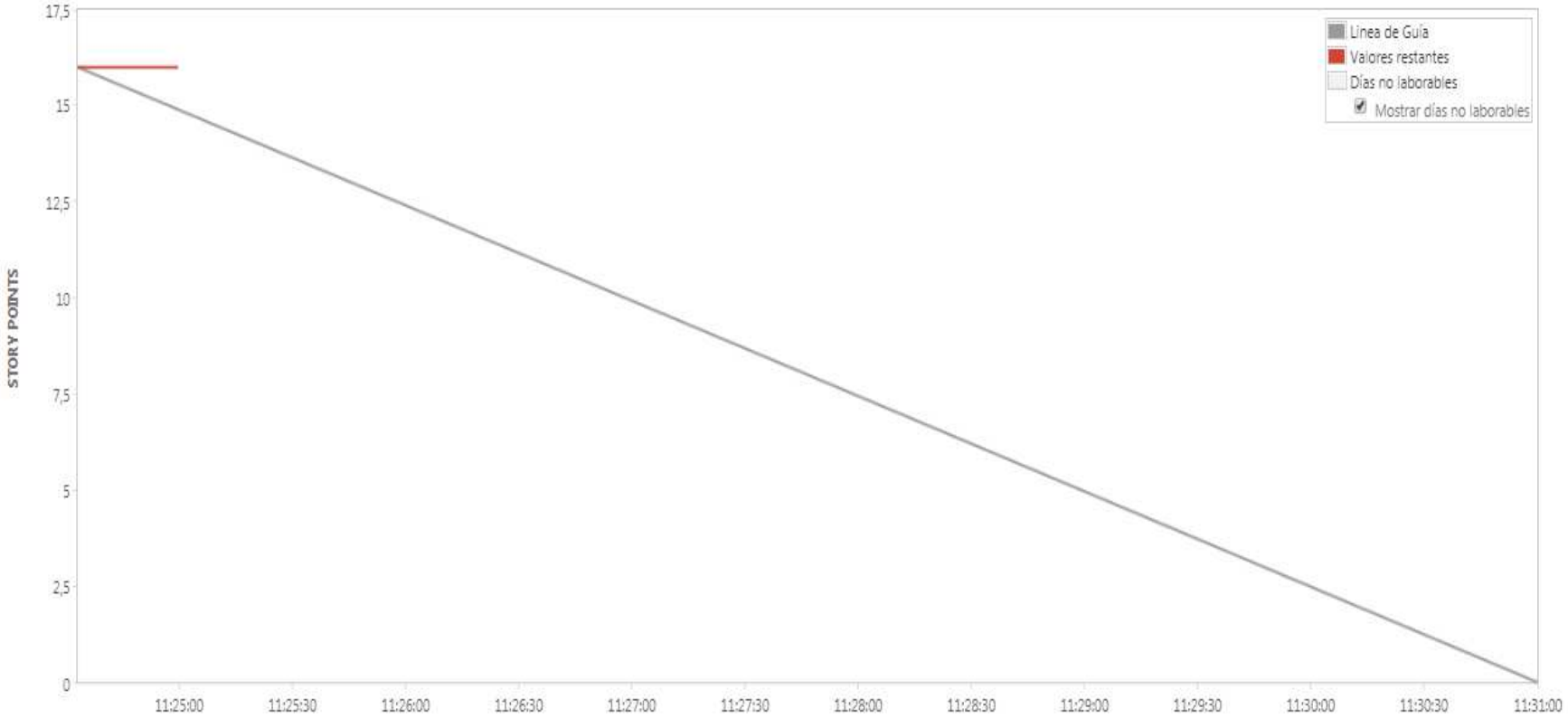
Se mantuvo una coordinación mutua con el jefe del área de patrimonio, o algún encargado de dicha oficina, con el cual se pudo solucionar dudas y errores en forma oportuna.

4.4.5. Equipo sentado juntos

Al estar en el ambiente de la oficina de Informática, hubo una buena comunicación, con lo cual se aprovechó la comprensión compartida y un mejor compromiso con el proyecto. Se procedió a elaborar el BurnDown chart para poder ver la velocidad de trabajo del equipo y también usarlo en los Scrum diarios

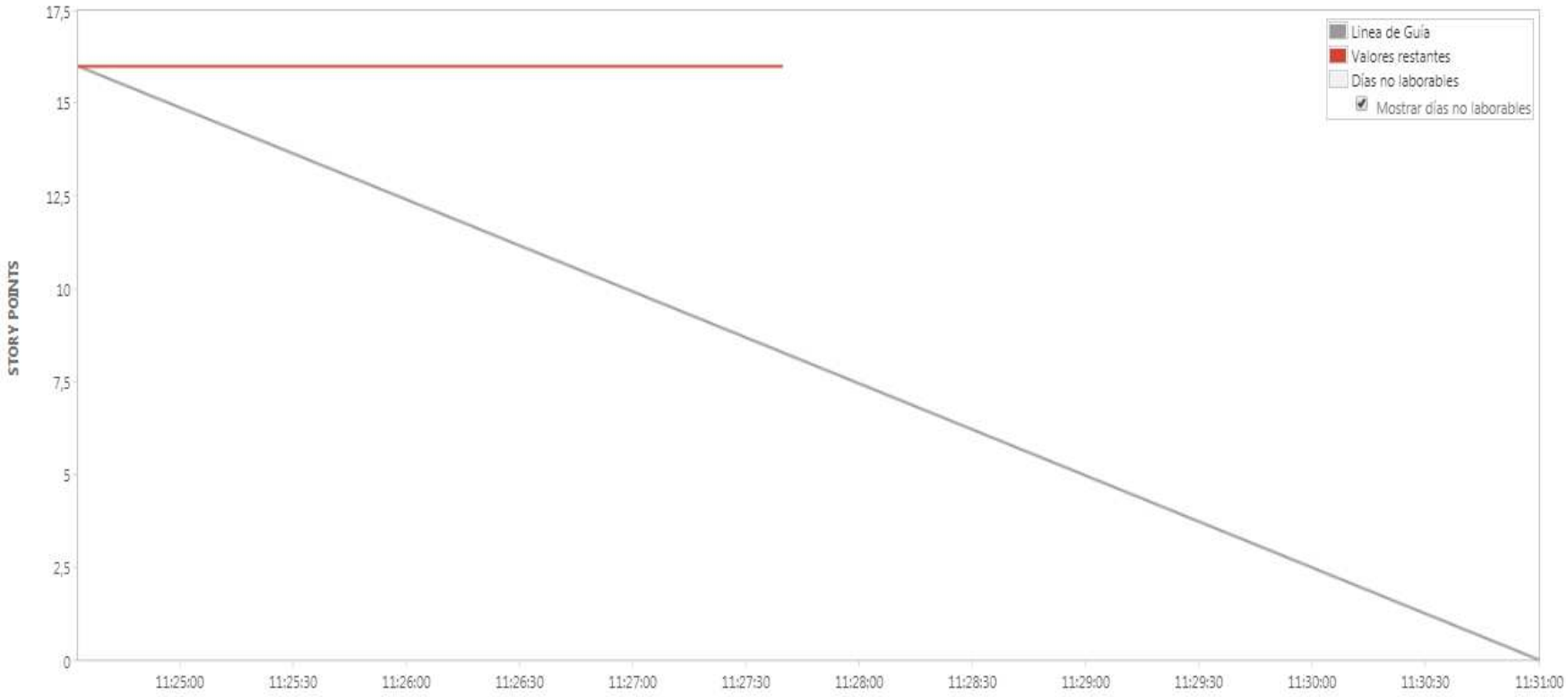
Proyectos / INVENTARIO-UGEL-HUAMANGA / Tablero IUH / Informes

Gráfica de trabajo pendiente



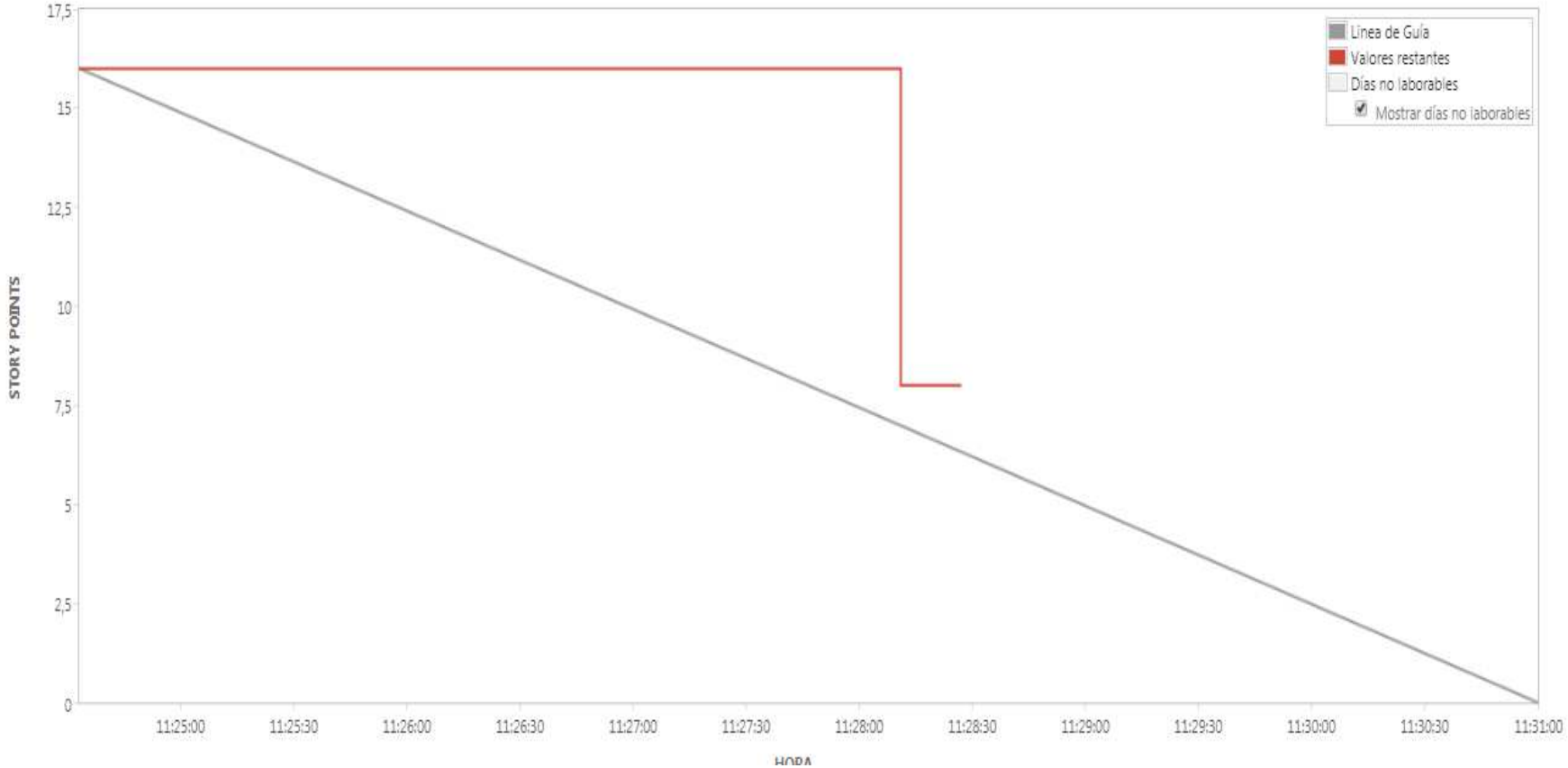
Proyectos / INVENTARIO-UGEL-HUAMANGA / Tablero IUH / Informes

Gráfica de trabajo pendiente



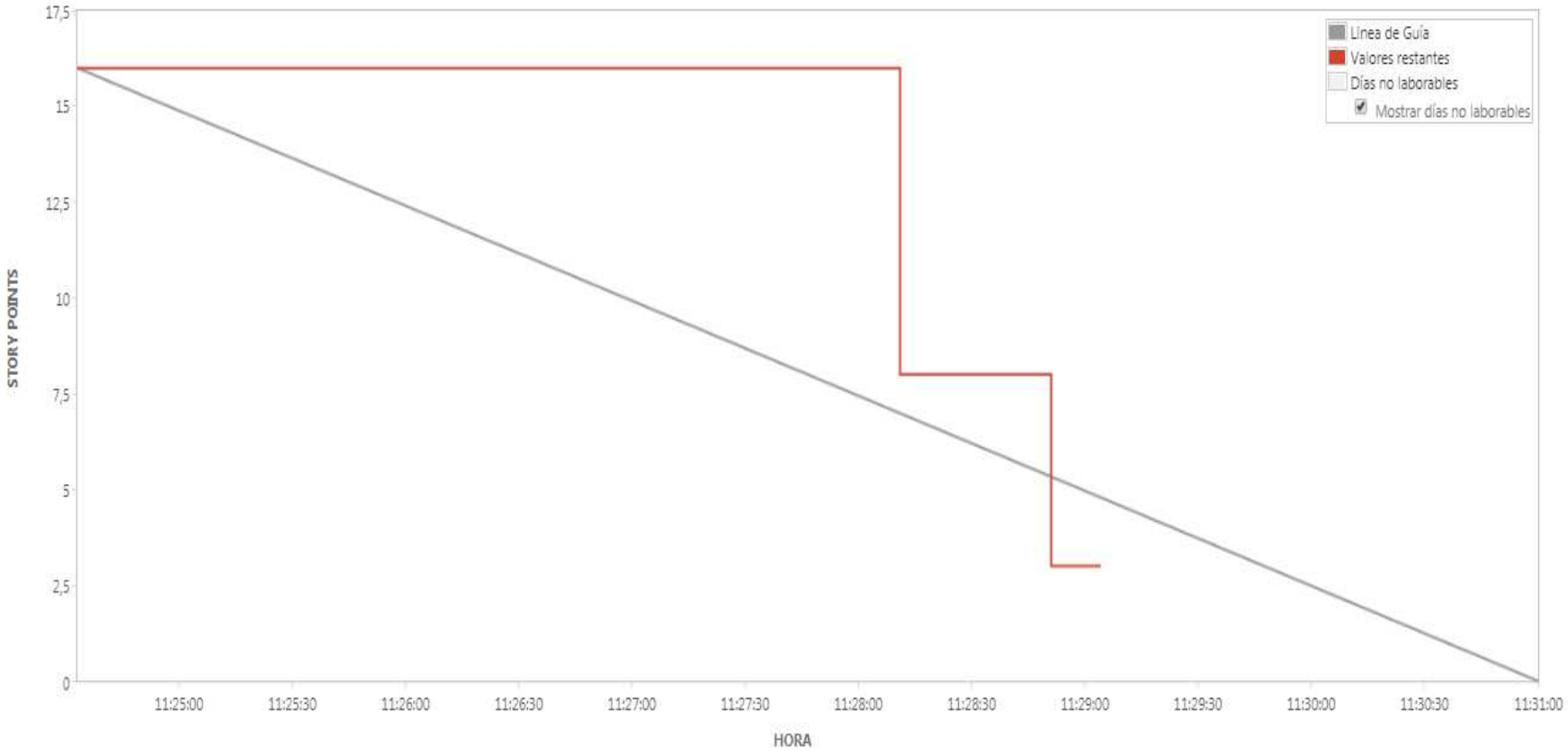
Proyectos / INVENTARIO-UGEL-HUAMANGA / Tablero IUH / Informes

Gráfica de trabajo pendiente



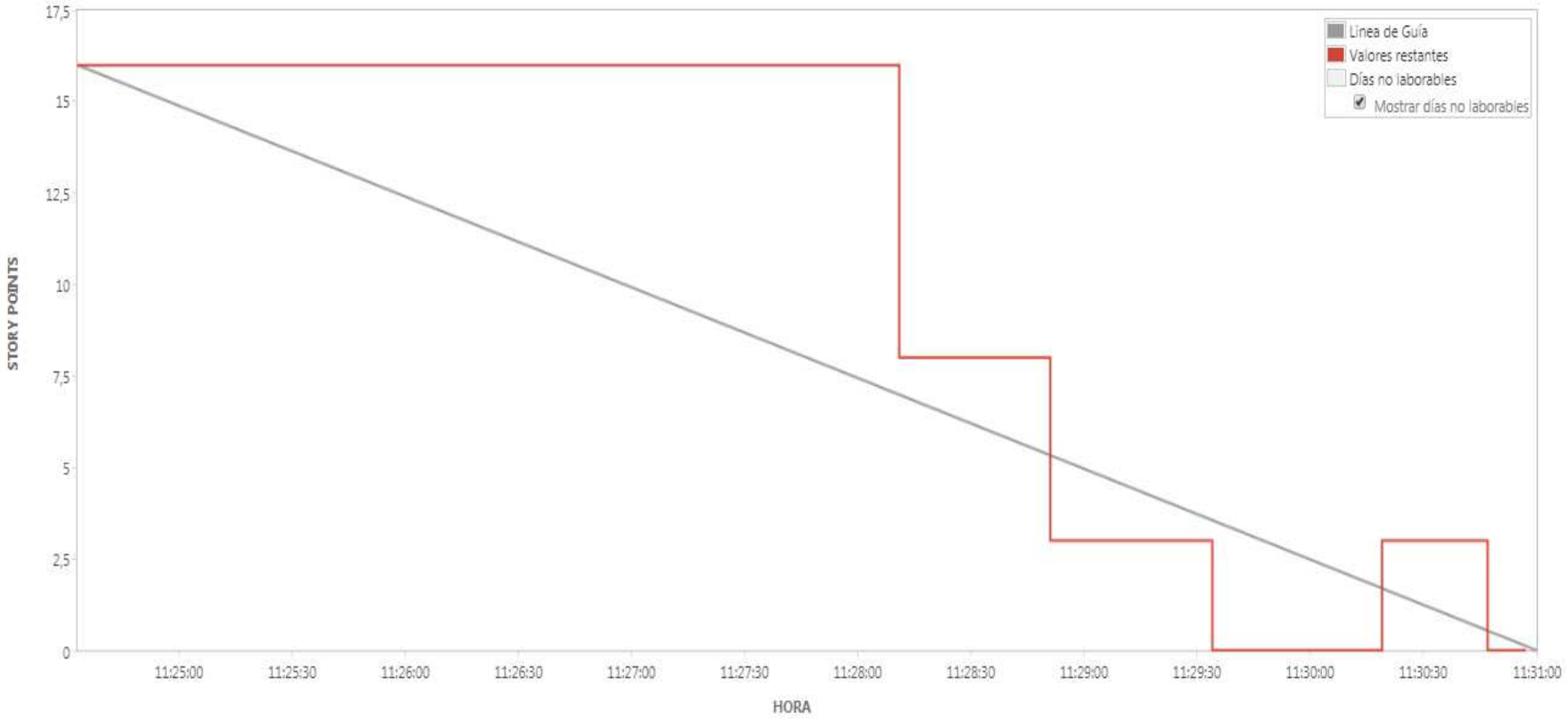
Proyectos / INVENTARIO-UGEL-HUAMANGA / Tablero IUH / Informes

Gráfica de trabajo pendiente



Proyectos / INVENTARIO-UGEL-HUAMANGA / Tablero IUH / Informes

Gráfica de trabajo pendiente



En las reuniones del Scrum diario se pudo constatar generalmente que se avanza tal cual se estuvo proyectado, salvo con uno que otro inconveniente propio de una institución pública, como actividades extra laborales, reuniones del personal, procesos de contrata y adjudicaciones, etc., gracias a la colaboración conjunto entre todos los stakeholders del proyecto se fue logrando cada objetivo. Se va revisando y actualizando los tableros, siendo el de mejor visualización el Tablero Kanban.

Para realizar las reuniones diarias se tomó el horario de las 10 de la mañana, en reunión con todos los miembros del equipo y el cliente que previamente ya están informados de los horarios, para la realización de toda esta primera iteración, con una duración promedio de 12 minutos, con exposiciones breves y comentarios de lo que se hizo, y planificar las tareas críticas para ese día con el fin de sincronizar esfuerzos en la realización de los trabajos. Se tratan 3 temas principalmente:

- Historias de usuario en trabajo.
- Pila del Sprint, gráficos de burndown y up y el tablero con actualizaciones diarias.
- Reporte de los miembros del equipo, que actualizarán sus actividades.

4.4.6. Programación en pareja

Juntos al responsable de la oficina de Informática, se pudo revisar tanto las interfaces, el código, revisiones, con lo cual se aceleraron tiempos y se mejoró la calidad.

4.4.7. Limitar el WIP

Es el mayor inconveniente que se tuvo en el desarrollo del proyecto, producto de las actividades propias de una institución pública, por lo cual las cartas Kanban se iban acumulando en las columnas del tablero, teniendo que dedicar más días en la realización del proyecto que fueron subsanadas durante el mismo.

4.4.8. Propiedad del código colectivo

A través de este principio, cualquiera de los miembros pudo arreglar partes de código y poder re factorizar, modificar diseños.

4.4.9. Normas de codificación

Mejores prácticas en la codificación.

```
pst.setString(1, txtCod_inventario.getText());
pst.setString(2, txt_denominacion.getText());
pst.setString(3, txtMarca.getText());
pst.setString(4, txtModelo.getText());
pst.setString(5, txtSerie.getText());
pst.setString(6, txtColor.getText());
pst.setString(7, cantidad);
pst.setString(8, fecharegistro);
```

```
pst.setString(1, TxtCodigo.getText());
pst.setString(2, TxtNombre.getText());
pst.setString(3, TxtApp.getText());
pst.setString(4, TxtApm.getText());
```

4.4.10. Pruebas

Se manejan las 3 pruebas:

Test unitarios:

Los TDD (Test Driven Development) son aquellas pruebas que tiene la finalidad de testear códigos individuales. Con el objetivo de comprobar la correcta estructura y que cumplen con su funcionalidad. Hubo componentes que no cumplían con la funcionalidad requerida, por lo que se procedió a rectificarlos. Algunas de las pruebas ejecutadas son:

PRUEBA	DESCRIPCIÓN	RESULTADO ESPERADO	STATUS	COMENTARIOS
01	Log in usuario	Función que genera una conexión	Correcto	Validación de credenciales de acceso del usuario, si es válido obtiene el acceso, de no ser validos los datos de conexión la función devuelve un mensaje de error.
02	Log out usuario	Función que genera el cierre de sesión.	Correcto	Esta función sirve para cerrar una sesión previamente generada.
03	executeUpdate	Función que actualiza un inventario más.	Correcto	Agrega un ítem más a la base de datos.
04	Cambiar datos del trabajador	Función que actualiza datos de un usuario registrado.	Correcto	Manejar excepciones si ocurriese un error.
05	Cambiar datos de equipo	Función que actualiza datos de un inventario registrado.	Correcto	Manejar excepciones si ocurriese un error.
06	Generar Listado de todos los equipos	Se lista todo el inventario registrado.	Correcto	Manejar excepciones si ocurriese un error.
07	Generar Listado de todos los equipos de un área	Se lista el inventario registrado por todos los usuario de un área.	Correcto	Manejar excepciones si ocurriese un error.
08	Generar reporte por	Se lista el inventario	Correcto	Manejar excepciones si ocurriese un error.

Test de aceptación:

Son test probados por el cliente para verificar que los resultados cumplan con sus expectativas, pero de manera gráfica mediante una interfaz ya que él no cuenta o desconoce del comportamiento de métodos, clases, etc. da calidad al producto y pues ayuda a los miembros del equipo cuan buenos fueron interpretando los requerimientos del cliente.

PROCESO	SUB PROCESO	DESCRIPCIÓN	ENTRADA	SALIDA	STATUS
Login	Ingreso de usuario y contraseña Cerrar una sesión	Validación de controles	Ingreso de usuario y contraseña	Pantalla principal	OK
Registro y modificación de inventario	Registro de inventario Cambiar datos de equipo	Interfaces donde se podrá registrar o modificar el inventario	Intentar registrar sin completar todo el formulario	Mensaje de error exigiendo que debe de ingresar todos los datos	OK
Reportes	Listado de todos los equipos, por área y usuarios	Interfaces para cada tipo de reporte con sus respectivos campos	Realizar un reporte con ayuda de los controles Descargar un reporte	Se obtiene el listado del inventario del reporte seleccionado Se descarga el reporte en formato	OK

				pdf.	

Test de integración:

Como producto de las liberaciones continuas, estas se deben de integrar al sistema cada vez que salgan, pero puede que haya problemas o errores al integrar, para eso es que sirven estos test comprobar la adecuada integración de la funcionalidad con el todo.

PRUEBA	DESCRIPCIÓN	RESULTADO ESPERADO	STATUS	COMENTARIOS
01	Login	Funcionamiento correcto de login, si el usuario es válido la aplicación debe de otorgar acceso al menú principal, si el usuario es invalido la aplicación debe de mostrar mensaje de error.	OK	Los campos usuario y contraseña son obligatorios. El mensaje de error debe de ser aceptado para desaparecer.
02	Módulo inventario	Funcionamiento correcto de todo el registro y modificación de	OK	Los campos deben ser llenados en su totalidad a pesar de no tener valor, de lo contrario el

		inventario, sin ningún error que pueda ocasionar la salida de la aplicación.		mensaje de error debe ser explícito, al igual si se quiere modificar un registro.
03	Modulo reporte	Funcionamiento correcto de todos los reportes, así como su posterior descarga.	OK	Establecer un mensaje de error si se quiere generar un reporte sin haber especificado el tipo.

Durante el proceso de pruebas se aplicará el re factoring de todo el código duplicado.

```
    pstl.setString(1, txtusuario.getText());
    pstl.setString(2, txtxontrasenia.getText());
JOptionPane.showMessageDialog(null, "Seguro que desea modificar usuario");
int n=pst.executeUpdate();
    int nl=pstl.executeUpdate();
```

```
    pstl.setString(2, txtxontrasenia.getText());
JOptionPane.showMessageDialog(null, "Seguro que desea modificar usuario");
int n=pst.executeUpdate();
    int nl=pstl.executeUpdate();

if(n>0){
        if(nl>0){
JOptionPane.showMessageDialog(null, "Registro Guardado con Exito");
        }
 }else {
JOptionPane.showMessageDialog(null, "Error al insertar el registro.", "Error", JOptionPane.ERROR_MESSAGE);
 }

} catch (SQLException ex) {
    Logger.getLogger(Modificar_Persona.class.getName()).log(Level.SEVERE, null, ex);
} catch (FileNotFoundException ex) {
    Logger.getLogger(Modificar_Persona.class.getName()).log(Level.SEVERE, null, ex);
}
```

4.4.11. Integración continua

Tomando en cuenta la complejidad, se procede Con la integración de los módulos en el sistema, para luego hacer las pruebas de integración para comprobar la correcta integración. Lo recomendable es empezar a integrar cada vez que el equipo de desarrollo suelte algún incremento y sea aprobado por el cliente.

Al final de la Iteración 1 se presenta un incremento de producto en cuanto al funcionamiento correcto del registro de inventario, registro y modificación de persona.

En las iteraciones siguientes se cumplió con los objetivos establecidos, con funcionalidades probadas y aprobadas.

4.4.12. Ritmo sostenible

Iteración 2

Se empezó la codificación en esta iteración 2, con las mejoras propuestas gracias a las reuniones diarias y a la retrospectiva, aplicando mejor las técnicas y prácticas de este modelo de marco híbrido. También se mejoró las pruebas gracias al equipo multidisciplinario y empoderado en la toma de decisiones.

El equipo gana experiencias en el desarrollo del proyecto es así que con la re factorización y la propiedad colectiva del código, se ayudan para mejorar el código, los tiempos y presentar los incrementos con calidad, con lo que hubo problemas gracias ya que hay personas reacias al cambio fue con el de trabajar en pares, ya que tiene aún la ideología de que si producen algo, ellos son los propietarios, se fue corrigiendo en esta iteración.

Scrum Diario Iteración 2

Se sigue con el modelo de la iteración anterior, pero aplicando mejoras, con retroalimentación continúa para no perder el objetivo del incremento, con constante intervención del cliente en el desarrollo, actualizaciones de los tableros, y el Scrum master estar al pendiente de que los equipos actualicen sus tareas y avances en estos tableros.

Retrospectiva Iteración 2

En la reunión, se pudo identificar aquellos puntos en los que los programadores destacaban en cumplir, también se identificó las mejoras y posibles problemas para la siguiente iteración, se cumplió con lo estipulado para la iteración, es decir el valor de 16 puntos.

Planificación de la siguiente iteración

Al juntar a todos los miembros del proyecto en un ambiente incluido el dueño del producto, para el análisis del sprint anterior, el cliente se mostró satisfecho por lo producido, para este nuevo inicio, se planteó las dudas que surgieron en el equipo de desarrollo, las cuales fueron absueltas por el cliente, se procedió a hacer un cambio mínimo en el Product Backlog en lo que se refiere a la priorización de funcionalidades que serían entregadas, al terminar la reunión todos entendieron cuál era el objetivo para esta nueva iteración

4.4.13. Pequeños lanzamientos

Lecciones aprendidas generales

Como cierre de proyecto, se preparó la documentación de errores para futuros proyectos, se perfeccionó y mejoró las técnicas de esta propuesta de metodología

híbrida para nuevos proyectos, el equipo de trabajo conoce ahora mejor las funciones que tienen en cada área, saben trabajar en pares.

Como último documento presentado en este cierre de proyecto se hizo el acta de entrega de producto con la satisfacción total del cliente, y un producto con valor para este.

4.5. RESULTADOS

Una vez aplicada esta propuesta de modelo de gestión ("XP + Scrum + Kanban") en este proyecto, hubieron resultados buenos y resultados por mejorar:

Con la encuesta es evidenció que la implantación de esta propuesta de marco de trabajo resultaría más fácil en instituciones que tienen mayor grado de conocimiento de metodologías ágiles. Y que por lo tanto aquellas instituciones con puntaje mejor a 60 tendría que trabajarse mucho más para poder implantar la metodología.

Al estar el Scrum master en constante guía y apoyo a los equipos del proyecto, se evidenció que estos, dependían de el para el avance, esperando que les indicase lo que debían de hacer, para las siguientes iteraciones se corrigió esto.

El compromiso y la responsabilidad por parte del equipo en el desarrollo de tareas y actualización del tablero causo algunos retrasos.

Debido a la decidía de parte de algunos miembros del equipo de trabajar en pares se tuvo la necesidad de mostrar con el ejemplo, para que diferencien las ventajas que tenía esta forma de trabajar, y lo que suponía para el desarrollo del proyecto en las siguientes iteraciones.

Gracias a la limitación del Wip se logró que el equipo de desarrollo se enfocase en una sola actividad, que los otros miembros del equipo que pertenecían a otros proyectos, re evaluarán sus prioridades, por lo que se mejoró los tiempos, recursos y horas hombre.

Con la adopción de esta propuesta de modelo de gestión se crearon equipos multi disciplinarios y auto organizados con una visión clara de roles y

responsabilidades, evidenciando mejoras constantes en cada tarea y actividad realizada.

Como parte del proceso de lanzamiento de mejoras continuas para el cliente, este se vio fuertemente beneficiado con funcionalidad de calidad que cumpla con sus expectativas, logrando ventaja competitiva contra otras empresas.

Las historias de usuario usadas en esta propuesta de modelo de trabajo ágil, al tener un alto grado de detalle y estar elaboradas en lenguaje del cliente, se obtuvo una comprensión casi total de las mismas, las que ayudaron en la creación de tareas que fueron desarrolladas por el equipo de programadores, se pudo priorizar el valor de las mismas gracias a las retroalimentaciones que brindaba el cliente, obteniendo resultados con calidad.

Gracias a la liberación de software potencialmente usable por el cliente, constante y continua, el cliente en diversas ocasiones pudo quedar satisfecho sin hacer ningún cambio al producto entregado, otras hubo que hacer modificaciones mínimas, pero en todas el cliente se mostró satisfecho con el desarrollo de su producto. También se debe en gran medida a la participación del cliente durante todo el proyecto, con las consultas de dudas, verificación de comportamientos, etc.

Las reuniones diarias contribuyeron enormemente con la retroalimentación de las historias de usuario que se vieron reflejadas en la calidad del incremento entregado al cliente. Cada uno de los miembros al ser auto organizado pudo consultar con total fluidez y sin demora con el cliente para que resolviera alguna duda, compartiendo experiencias y conocimientos del sistema.

Siendo los equipos de trabajo, equipos dinámicos, con flexibilidad, se llegaron a formar equipos multidisciplinarios de acuerdo a las actitudes y aptitudes de los mismos, que al tener un buen ambiente laboral, se notó la mejora de los tiempos y la calidad del producto que ofrecían.

Como buena práctica y aunque no estuvo contemplada en las prácticas de la propuesta del marco de trabajo, el documentar las experiencias aprendidas,

resultaron ser de ayuda en la resolución de problemas, resultó ser una guía de consulta para los miembros del equipo.

En las reuniones de retrospectiva, se aplicó la autocrítica con el fin de mejorar los procesos de los equipos de trabajo, al tener claras las deficiencias también se tenían claras las soluciones.

Como parte de las prácticas de esta propuesta de marco de trabajo, se tuvo la intervención del cliente 24/7 durante todo el proyecto, es así que obtuvo conocimiento del sistema, y los equipos obtuvieron conocimiento de su empresa y mejoraron la intensión de los entregables para cumplir con las expectativas.

Líneas arriba se lee la máxima intervención del cliente en el proyecto, es así que hubo un conocimiento reciproco de los actores, comprendiendo así mejor las necesidades del otro, que se transformó en cliente satisfecho con producto de calidad, esta relación fue armonizando mejor con el pasar de las iteraciones. El cierre del proyecto resulto mucho mejor de lo esperado.

Como parte de utilizar metodologías enfocadas en la organización y flujo de trabajo, los tiempos y entregas se cumplieron, el trabajo global se organizó mejor, se observó siempre el flujo y evolución del proyecto, con actualización constante de los cuadros de control y toma de decisiones como los Burn Down.

Gracias al uso de las prácticas de Xp se han obtenido muy buenos resultados en la calidad del código por el uso de pruebas, siendo estas:

Metodología enfocada a la programación, por lo que el producto resultó ser de calidad, gracias a las pruebas unitarias, de aceptación e integración, disminuyendo los errores y tiempos, con una aceptación y satisfacción máxima del cliente que se vio reflejada en el producto de valor que utiliza ahora en su institución, con un entorno amigable y con las funcionalidades necesarias para su uso.

El uso de tarjetas CRC ha ayudado a los programadores en mapear la relación de las clases y sus métodos, mayor facilidad de programación, aceleración de tiempos, disminución de recursos y la prioridad de las historias de usuario.

Diseño simple, fácil de comprender, hacer lo mínimo para pasar la prueba, el uso del re factoring, programación en pares son características que han incrementado la reducción de errores, disminución de recursos, obtención de producto de calidad y satisfacción del cliente.

El invertir en programación por parejas es notablemente beneficioso a mediano plazo, debemos de cambiar la mentalidad de aquellas personas que no querían el cambio en la empresa, señalarles que los cambios son para mejorar la empresa, para mejorar como institución, que permitirán ahorrar recursos, cometer menos errores, codificar mejor.

Surge la duda de, qué pasaría si se hubiese trabajado con las metodologías por separado:

Si el proyecto se hubiese realizado netamente con la metodología Xp, el producto seguramente tendría mucha mejor calidad, se hubiese logrado comprender mejor el trabajo en pares sin tener tantas trabas y hubiésemos tenido código mucho más libre, sin propiedad, pero al fijarnos en la organización, en la gestión de los procesos, hubiésemos tenido más retrasos, menos interacción con el cliente, menos satisfacción de sus necesidades, que para la empresa hubiese sido una opinión negativa y para el cliente menos valor a su producto.

Si sólo se usó Scrum, tendríamos un mejor listado de historias de usuario, mejor detalladas, con requerimientos exactos, con sprint tradicionales, con seguimiento y controles importantes, pero en cuanto a calidad del producto, calidad de código, sin someterlos a las pruebas, sería un producto deficiente sin cumplimiento de expectativas.

Si se hubiese realizado únicamente con Kanban, se hubiese tenido una mejor forma de ver las tareas de todo el proyecto, aquellas por hacer, aquellas en proceso y aquellas ya realizadas, así como el limitar la cantidad de tareas en las que trabaja el equipo simultáneamente, consiguiente un mayor control de las fases del proyecto, flexibilidad y disminución del tiempo dedicado a tareas poco productivas, pero hubiésemos tenido un resultado de poca calidad, además de reducir la interacción del equipo.

Al integrar entonces estas tres metodologías ágiles y plantear este modelo de desarrollo híbrido, se obtuvo un código de calidad, un producto acorde a las necesidades del cliente, gracias a que el cliente fue un actor más dentro del proyecto, mucho mejor organización y gestión de los procesos de desarrollo, gracias a las reuniones se tuvo mejores retroalimentaciones, y el flujo de trabajo global de manera clara ayudo a todos los miembros del equipo a saber dónde se estaba y hacia donde se quería llegar. Por tanto al complementarse estas tres metodologías se logra cumplir sino al 100% con las necesidades del cliente dando valor al producto.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

a) De acuerdo al numeral 4.6, del Capítulo IV – Resultados, se comprueba que el modelo de gestión de desarrollo de software ágil mediante Scrum y Kanban se adapta al modelo iterativo e incremental de la Programación Extrema.

b) De acuerdo al numeral 4.6, del Capítulo IV – Resultados, se comprueba que el modelo de gestión de desarrollo de software ágil mediante Scrum y Kanban se adapta a las pruebas unitarias de la Programación Extrema.

c) De acuerdo al numeral 4.6, del Capítulo IV – Resultados, se comprueba que el modelo de gestión de desarrollo de software ágil mediante Scrum y Kanban se adapta a la refactorización de código de la Programación Extrema.

d) De acuerdo al numeral 4.6, del Capítulo IV – Resultados, se comprueba que el modelo de gestión de desarrollo de software ágil mediante Scrum y Kanban se adapta a la corrección de errores y código compartido de la Programación Extrema.

5.2. RECOMENDACIONES

a) Se recomienda la aplicación de esta propuesta de modelo de gestión de trabajo en distintos tipos de trabajos y proyectos para poder tener una mejor valorización de ventajas y contras.

b) Especializar a los trabajadores en funciones y roles específicos, que generaran aún más valor y calidad al producto, con una disminución en el tiempo de las iteraciones.

c) Mientras que el desarrollo de nuevos proyectos estén inmersos en metodologías ágiles, se recomienda el uso de este modelo no importando el tamaño del proyecto.

d) Trabajar en el material humano, mediante capacitaciones y talleres, un empleado mejor motivado trabaja mucho más que uno que sólo viene a cumplir con la empresa.

e) Al principio de la utilización de esta propuesta de modelo de gestión de trabajo, puede provocar una diversidad de falencias, con lo que se querrá renunciar a la implantación del marco de trabajo y regresar a las metodologías tradicionales; sin embargo, se recomienda ser perseverante, no rendirse ya que ayudará a la Institución, con una mejor experiencia documentada para futuros proyectos y equipos.

REFERENCIA BIBLIOGRÁFICA

Abrahamsson, P., Salo, O., Ronkainen, J. y Warsta, J. (2002). *Agile Software Development methods: Review and Analysis.*

Agilemanifesto (2019). *Agilemanifesto.* Recuperado el 21 de febrero de 2020, de: https://agilemanifesto.org/iso/es/principles.html.

Ahmad, Z. (2014). *Scrumban - Adaptive Agile Development Process. Using Scrumban to improve software development process*. Tesis de maestría. Helsinki Metropolia University of Applied Sciences. Finlandia. Disponible:https://www.theseus.fi/bitstream/handle/10024/77014/Khan_Zahoor.pdf? sequence=1

Bahit, E. (2012). *Scrum y Extreme Programming*. Argentina, Buenos Aires.

Ballesteros, D. "*A practical form to apply the System Kanban in the Colombian Mypimes*". Scientia et Technica Año XIV, N.° 39. 2008.

Balza, L. M., Briceño, T., Linares, B., Lobo, R., & Nuñez, C. *INFORME SOBRE XP*. Universidad Valle del Momboy. 2007

Beck, K. y Andrés, C. (2005). *Extreme Programming Explained: Embrace Change*. USA, Boston: Addison- Wesley.

Bermejo, M. (2011). *El Kanban*. Barcelona, España: UOC.

Canos, J., Leteller, P. y Penades, C. (2004). *Metodologías Agiles en el Desarrollo de Software*. Valencia, España.

Carmichael, A. y Anderson, D. (2015). *Essential Kanban Condensed.*

Carrasco, M., Ocampo, W., Ulloa, L. y Azcona, J (2019). *Metodología híbrida de desarrollo de software Combinando XP y Scrum*. Mikarimin, 5(2),109-116.

Casas, S., Reinaga, H. (2009). *Aspectos tempranos: Un enfoque basado en tarjetas CRC.* Sociedad Colombiana de Computación. https://www.researchgate.net/publication/220136724_Aspectos_tempranos_Un_enfoque_basado_en_tarjetas_CRC

Chiluisa Pallo, A. P., & Loarte Cajamarca, B. G. (2014). *Desarrollo e Implantación del Sistema de Control de Inventarios y Gestión de Laboratorios para la Facultad de Ciencias de la Escuela Politécnica Nacional*. Quito.

Cifuentes Lozano, A. (2012). *Integración de Buenas Prácticas.* Recuperado el 24 de Noviembre de 2014, de: http://bibliotecadigital.icesi.edu.co/biblioteca_digital/bitstream/10906/68430/5/modelo_integracion_practicas.pdf

Cohn, M. (2006). *Planning Poker*. Mountain Goat Software, 56-59.

Córdova, M. (2003). *Estadística Descriptiva e Inferencial*. Perú, Lima: Moshera S.R.L.

Deemer, P., Benefield, G., Larman, C., & Bas, V. (2009). *The Scrum Primer Versión en Español.* California, Estados Unidos.

Deemer, P., Benefield, G., Larman, C., & Bas, V. (2009). Información Básica de Scrum the Scrum Primer Version 1.1. Scrum Training Institute. Traducción de Leo Antoli. Agile-Spain. Disponible en: http://www. goodagile.com/scrumprimer/scrumprimer_es.pdf. Consulta: 30 de mayo del 2011.

DeMarco, T. y Boehm, B. (2002). *The agile methods fray*. United States, New York.

Díaz, R. (2009). *Las metodologías agiles como garantía de calidad de software.* España.

Du Bois, B., Demeyer, S., & Verelst, J. (2004, November). *Refactoring-improving coupling and cohesion of existing code. In 11th working conference on reverse engineering*.IEEE.144-151.

Fernández, J. (2014). *Las Metodologías Agiles*. España, Barcelona.

Ferreira Escutia, R. (2013). *XP Extreme Programming*. Recuperado el 2015, de http://slideplayer.es/slide/84721/

Fitsilis, P. (2008). *Comparing PMBOK and Agile Project Management Software Development Processes in Advances Computer and Information Sciences and Engineering*. Netherlands: Springer.

Gamboa, J. (2014). *Aumento de la Productividad en la Gestión de Proyectos, Utilizando una Metodología Ágil Aplicada en una Fábrica de Software en la Ciudad de Guayaquil.* Tecnológica Espol. 27(2), 1-36.

Haugen, N. C. (2006, July). *An empirical study of using planning poker for user story estimation*. In AGILE 2006 (AGILE'06). IEEE.9.

Hernández, R., Baptista, P. y Fernández, C. (2014). *Metodología de la Investigación.* México: McGraw-Hill/ Interamericana Editores.

Jeffries, R., Anderson, A. y Hendrickson, C. (2001). *Extreme Programming Installed*. USA: Addison-Wesley.

Joskowicz, J. (2008). *Reglas y prácticas en eXtreme Programming*. Universidad de Vigo, 22.

Kanbanize (2020). *Kanbanize*. Recuperado el 3 de enero de 2020, de: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-tablero-kanban/

Klipp, P. (2014). *Getting Started with Kanban.*

Kniberg, H. (2007). *Scrum y XP desde las trincheras*. USA: C4Media Inc.

Ladas, C. (2008). *Scrumban Essays on Kanban Systems for Lean Software Development.* Disponible en: https://books.google.co.ve/books?hl=es&lr=&id=SQFdAgAAQBAJ&oi=fnd&pg=PA7&dq=Scrumban-Essays+on+Kanban+Systems+for+Lean+Software+Development&ots=c89XRBXGNg&sig=I59JUW5uUe2KDdWgc-LuJnkQKd8#v=onepage&q=Scrumban-Essays%20on%20Kanban%20Systems%20for%20Lean%20Software%20Development&f=false

Laudon, K. y Laudon, J. (2004). *Sistemas de Información Gerencial*. México: Pearson Educación.

Leiva, I., & Villalobos, M. (2015). Método ágil híbrido para desarrollar software en dispositivos móviles. *Ingeniare. Revista Chilena de Ingeniería*, 473-488.

Letelier, P., & Penadés, M. C. (2006). Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Ciencia y Técnica Administrativa*.

Letelier, P., & Penadés, M. C. (13 de Marzo de 2012). *Repositorio institucional de la Universidad de Las Tunas.* Obtenido de http://roa.ult.edu.cu/bitstream/123456789/476/1/TodoAgil.pdf

Martel, A. (2018). *Antonio Martel Blog.* Recuperado el 9 de diciembre de 2018, de: https://www.antoniomartel.com/2018/12/los-valores-de-extreme-programming.html.

Martínez, J. (2016). *Fundamentos de Programación en Java.* España, Madrid: EME.

Monroy, S. (2008). *Estadística Descriptiva*. México.

Moré Martín, A. (2011). *MPlu+ a Ágil: el modelo de proceso centrado en el usuario como metodología ágil*. Recuperado el 9 de febrero de 2011, de: https://repositori.udl.cat/handle/10459.1/45841.

Ohmyroot!. *Estándares de codificación.* Recuperado el 12 de enero de 2017, de: https://www.ohmyroot.com/buenas-practicas-legibilidad-del-codigo/

Palacio, J. (2007). *Flexibilidad con Scrum Principios de diseño e implantación de campos de Scrum*. Safe Creative.

Pardo, M. y Murado, E. (2010). *Metodologías de desarrollo agiles: Scrum y Extreme Programming.*

Pérez, M. (2014). *Guía comparativa de metodologías ágiles.* Tesis de Grado en Ingeniería Informática de servicios y aplicaciones no publicado. España: Universidad Valladolid. Recuperado el 17 de febrero de 2016, de: https://uvadoc.uva.es/bitstream/10324/1495/1/TFG-B.117.pdf.

Pérez, M. (2014). *Lenguajes de Programación Orientada a Objetos*. USA. CreateSpace Independent Publis.

Pressman, R. (2011). *Ingeniería del software*. United States, New York: The McGraw-Hill.

Pressman, R. (2010). *Ingeniería del Software. Un enfoque práctico.* México DF: Mc Graw Hill.

Richter, T. (2011). *Personal Kanban stop wasting your life.*

Robles, G., & Ferrer, J. (2002). *Programación eXtrema y Software Libre*. Universidad Politécnica de Madrid. España.

Rosas, A.K. (2017). *Sistema web para control de inventarios y rendimientos*. Repositorio Institucional de Investigación. Universidad Tecnológica del Centro de Veracruz, Recuperado el 4 de abril de 2017, de: http://reini.utcv.edu.mx/handle/123456789/239.

Rubin, K. (2013). *Essential Scrum*. USA: Pearson Education.

Salamon, A., Maller, P. A., Boggio, A., Mira, N., Perez, S., & Coenda, F. (2014). *La integración continua aplicada en el desarrollo de software en el ámbito científico–técnico*. In XX Congreso Argentino de Ciencias de la Computación (Buenos Aires, 2014).

Saliu, M. O., & Ruhe, G. (2007, September). *Bi-objective release planning for evolving software systems*. In Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (pp. 105-114).

Schwaber, K. y Sutherland, J. (2017). *The Scrum Guide*.

Shore, J. y Warden, S. (2008). *The art of Agile Development*.

Silva, M., Barrera, A., Arroyave, J. y Pineda, J. (2007). *Un método para la trazabilidad de requisitos en el Proceso Unificado de Desarrollo.*

Sommerville, I. (2010). *Ingeniería de software*. España, Madrid: Pearson Education Limited.

Supo, J. (2012). *Seminarios de Investigación Científica.* Arequipa:

Tesei, F., Cabrera, M., Vaquero, M., & Tedini, D. (2019). *Acercando la academia al mundo real: una experiencia de aplicación de Metodologías Agile al proceso de enseñanza-aprendizaje en una asignatura de desarrollo de software*. In I Simposio Argentino de Educación en Informática (SAEI 2019)-JAIIO 48 (Salta).

Tuya, J., Ramos, I., Dolado, J. (2007). *Técnicas Cuantitativas para la Gestión en la Ingeniería del Software*. España, La Coruña: Netbiblo S.L.

Vlaanderen, K., Jansen, S., Brinkkemper, S. y Jasper, E. (2011). *The agile requirements refinery: Applying SCRUM principies to software product management. Information and Software Technology*, 53(1),58-70 doi:10.1016/j.infsof 2010.08.004.

Wells, D. (2019). *Programación extrema: una introducción suave*. Recuperado el 28 de diciembre de 2019, de: http://www.extremeprogramming.org/values.html.

ANEXOS

Anexo A. Plantilla de Product Backlog

Id de la historia	Enunciado de la historia	Estado	Iteración (Sprint)	Prioridad	Comentarios
XXXX	Rol – descripción de la funcionalidad - finalidad	• Por hacer • En proceso • Terminada	X	Baja Media Alta	……

Anexo B. Plantilla de Historia de Usuario

HISTORIA DE USUARIO	
Número:	**Nombre de la historia:**
Usuario:	**Prioridad:**
Descripción:	
Observaciones:	

Anexo C. Plantilla de Tarjeta CRC

NOMBRE DE LA CLASE	
RESPONSABILIDADES	**COLABORADORES**

Anexo D. Plantilla de Task Card

TAREA:	
N° DE TAREA:	**N° DE HISTORIA:**
TIPO DE TAREA	
FECHA INICIO:	**FECHA FIN:**
PROGRAMADOR RESPONSABLE	
DESCRIPCIÓN	

Anexo E. Encuesta de factibilidad para aplicar una metodología

La presente encuesta es parte del trabajo de titulación de la carrera de Ingeniería de Sistemas de la Universidad Nacional de San Cristóbal de Huamanga. La información consignada en este documento será mantenida confidencialmente y únicamente se utilizaran los datos estadísticos totales con fines académicos.

Datos de la Empresa Encuestada

Nombre de la Empresa:……...

Ciudad:…….. **Cargo:…….**

Email de contacto: ……

Fecha:…….

¿Número de empleados de su empresa?

1-5	
6-10	
11-20	
21-49	
50 a más	

Cuestionario

1. ¿Los Ingenieros/Programadores, trabajan en varios proyectos en forma simultánea a tiempo exclusivo?

	Nunca
	Casi nunca
	A veces
	Casi siempre
	Siempre

2. ¿Cuál considera que es el nivel de aplicación de metodologías de gestión de desarrollo en su empresa?

	Muy bajo
	Bajo
	Regular
	Aceptable
	Muy alto

3. ¿Está establecida la política de la calidad y los objetivos de la calidad?

	Prácticamente no se realiza
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

4. ¿Están definidas las responsabilidades y autoridad entre ellas la función de calidad?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

5. ¿Los roles están bien definidos y limitan o aumentan las posibilidades de las personas?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

6. ¿Los proyectos anteriores han sido entregados en tiempo y forma o han sufrido retrasos?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

7. ¿Asegura la dirección la disponibilidad de los recursos necesarios: Humanos, instalaciones y equipos?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

8. ¿Se tienen identificados los requisitos de los clientes tanto los especificados por ellos como los no especificados, así como los requisitos legales y reglamentarios?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

9. ¿Se revisan los requisitos del producto o servicio antes de adquirir un compromiso con el cliente?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

10. ¿El cliente final está contento con el producto?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

11. ¿Se realizan planes para el personal (admisión, formación, desarrollo, etc.) evaluando el rendimiento y las necesidades de desarrollo de todas las personas?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

12. ¿Es prioridad satisfacer al cliente mediante la entrega temprana y continua de software con valor?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

13. ¿Se entrega software funcional frecuentemente?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

14. ¿Se acepta que los requerimientos cambien incluso en etapas tardías del desarrollo?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

15. ¿Los proyectos se desarrollan en torno a individuos motivados?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

16. ¿El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

17. ¿El software funcionando es la medida principal de progreso?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

18. ¿Los promotores, desarrolladores y usuarios son capaces de mantener un ritmo constante de forma indefinida?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

19. ¿Se analizan e interpretan las métricas y datos obtenidos?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

20. ¿El equipo de desarrollo y jefes de producto están físicamente en la misma área?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

21. ¿Los desarrolladores conocen la visión completa del producto?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

22. ¿Las decisiones de cambio son discutidas con el equipo de desarrollo o tomadas por la parte gerencial?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

23. ¿Existe rotación de las personas y roles o cada individuo trabaja en su especialidad?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

24. ¿Las personas pueden elegir su trabajo o tareas basadas en sus conocimientos o les son asignadas?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

25. ¿Su empresa Realiza pruebas de software?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

26. ¿Existe un programa de mejora continua que afecta a todas las actividades de la empresa empleando herramientas adecuadas y estableciendo objetivos de mejora?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

27. ¿Cuáles considera que son las principales dificultades en la aplicación de estándares de control de calidad?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

28. ¿Se controlan las no conformidades y se asegura que el producto no conforme es identificado y controlado para prevenir una utilización o entrega no intencionada?

	Prácticamente no
	Se realiza en ocasiones puntuales
	Se realiza en la mayoría de casos
	Se realiza en casi todas las áreas
	Se realiza siempre y de forma total

	1	2	3	4	5
Total marcados					
	x1	x2	x3	x4	x5
Suma total de puntos					

Anexo F. Fichas bibliográficas

FICHA BIBLIOGRÁFICA	
Libro	Scrum y Extreme Programming
Autor	Eugenia Bahit
Edición	Auto edición
Publicación	Argentina
Editorial	Safe Creative
Año	2012

FICHA BIBLIOGRÁFICA	
Libro	Essential Kanban Condensed
Autores	Andy Carmichael, David J. Anderson
Edición	Volumen 2
Publicación	Argentina
Editorial	Blue Hole Press
Año	2015

FICHA BIBLIOGRÁFICA	
Tesis	Modelo de integración de buenas prácticas para la gestión de proyectos de desarrollo de software para empresas donde dichos proyectos no son su objetivo de negocio
Autor	Adriana Y. Cifuentes Lozano
Fecha	23 de mayo de 2012
Publicación	Colombia

Editorial	Universidad Icesi
Fichero	https://repository.icesi.edu.co/biblioteca_digital/bitstream/10906/68430/1/cifuentes_modelo_integracion_2012.pdf

FICHA BIBLIOGRÁFICA	
Página	Principios del Manifiesto Ágil
Autor	Agilemanifesto
Fecha	2019
Fichero	https://agilemanifesto.org/iso/es/principles.html

FICHA BIBLIOGRÁFICA	
Libro	The Scrum Primer Versión en Español
Autores	Pete Deemer, Gabrielle Benefield, Craig Larman y Bass Vodde
Versión	1.1
Publicación	Estados Unidos
Editorial	Scrum Training Institute
Año	2009
Fichero	http://www.goodagile.com/scrumprimer/scrumprimer_es.pdf

FICHA BIBLIOGRÁFICA	
Libro	Reglas y prácticas en eXtreme Programming
Autor	José Joskowicz
Publicación	España
Editorial	Universidad de Vigo

Año	2008

FICHA BIBLIOGRÁFICA	
Página	Estándares de codificación
Autor	Ohmyroot!
Fecha	12 de enero de 2017
Fichero	https://www.ohmyroot.com/buenas-practicas-legibilidad-del-codigo/

FICHA BIBLIOGRÁFICA	
Libro	Flexibilidad con Scrum Principios de diseño e implantación de campos de Scrum
Autor	Juan Palacio
Publicación	España
Año	2007

FICHA BIBLIOGRÁFICA	
Libro	Ingeniería del software
Autor	Ian Sommerville
Edición	9na edición
Publicación	España
Editorial	Pearson Education
Año	2010

Anexo G. Ficha de análisis documental

FICHA DE ANÁLISIS DOCUMENTAL	
Nombre del documento (registre el nombre o título del documento consultado)	
Autor (registre el nombre completo del autor o autores del documento)	
Referencia bibliográfica según norma APA (registre la referencia bibliográfica completa de acuerdo a la estructura que corresponda en normal APA)	
Palabras clave de búsqueda (registre las palabras con las que realizó la búsqueda de cada documento)	
Palabras claves del texto (registre las palabras claves que aparecen en éste, en caso tal que no las tenga, se deja espacio en blanco o se escribe: No tiene)	
Ubicación (dirección electrónica específica) y/o clasificación topográfica de la biblioteca donde se encuentra (registre la URL para documentos encontrados en la red, o los datos correspondientes para documentos consultados en físico, como por ejemplo en bibliotecas, centros de documentación, entre otros, de acuerdo con las normas APA)	
Descripción del aporte al tema seleccionado (presente una descripción argumentada de aportes que considera pertinentes para el tema seleccionado, de acuerdo con lo que plantean el o los autores)	
Conceptos abordados (conceptos claves que le aporta a su tema explicando el por qué)	

UNSCH FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

"Año de la Universalización de la Salud"

ACTA DE SUSTENTACIÓN DE TESIS N° 018-2020-FIMGC

En la ciudad de Ayacucho, en cumplimiento a la **Resolución Decanal N° 141-2020-FIMGC-D**, siendo los catorce días del mes de setiembre del 2020, a horas 10.00 a.m.; se reunieron los jurados del acto de sustentación, en el Auditórium virtual google meet del Campus Universitario de la Universidad Nacional de San Cristóbal de Huamanga.

Siendo el Jurado de la sustentación de tesis compuesto por el Presidente el **Mg. Ing. Manuel A. LAGOS BARZOLA**, Jurado la **Mg. Ing. Celia Edith MARTÍNEZ CÓRDOVA**, Jurado el **Mg. Ing. Hubner JANAMPA PATILLA** y Secretario del proceso **Ing. Christian LEZAMA CUELLAR**, con el objetivo de recepcionar la sustentación de la tesis denominada **"MODELO DE GESTIÓN DE DESARROLLO DE SOFTWARE ÁGIL MEDIANTE SCRUM Y KANBAN SOBRE LA PROGRAMACIÓN EXTREMA, 2019",** sustentado por el Bach. **Eder SOLÓRZANO HUALLANCA, bachiller en Ingeniería de Sistemas;** siendo el Asesor de la tesis el **Mg. Ing. Hubner JANAMPA PATILLA.**

El Jurado luego de haber recepcionado la sustentación de la tesis y realizado las preguntas, el sustentante al haber dado respuesta a las preguntas, y el Jurado haber deliberado; califica con la nota aprobatoria de **18 (dieciocho).**

En fe de lo cual, se firma la presente acta, por los miembros integrantes del proceso de sustentación.

UNIVERSIDAD NACIONAL DE SAN CRISTOBAL DE HUAMANGA
ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS
Mg. Manuel A. Lagos Barzola
DIRECTOR

Firmado digitalmente por: CAcert WoT User
Fecha y hora: 14.09.2020 11:31:16

Mg. Ing. Manuel A. LAGOS BARZOLA
Presidente

Mg. Ing. Celia Edith MARTÍNEZ CÓRDOVA
Jurado

Mg. Ing. Hubner JANAMPA PATILLA
Jurado - Asesor

Ing. Christian LEZAMA CUELLAR
Secretario del Proceso

c.c.:
Bach. Eder SOLÓRZANO HUALLANCA
Jurados (4)
Archivo

UNSCH FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

CONSTANCIA DE ORIGINALIDAD DE TRABAJO DE INVESTIGACIÓN

El que suscribe; responsable verificador de originalidad de trabajos de tesis de pregrado en segunda instancia para las Escuelas Profesionales de la Facultad de Ingeniería de Minas, Geología y Civil; en cumplimiento a la Resolución de Consejo Universitario N° 039-2021-UNSCH-CU, Reglamento de Originalidad de Trabajos de Investigación de la UNSCH y Resolución Decanal N° 158-2021-FIMGC-UNSCH-D, deja constancia que:

- Apellidos y Nombres del Bach. : Solórzano Huallanca Eder
- Escuela Profesional : Ingeniería De Sistemas
- Título de la Tesis : Modelo de gestión de desarrollo de software ágil mediante SCRUM y KANBAN sobre la programación extrema, 2019.
- Evaluación de la originalidad : 8 % de similitud

Por tanto, según los artículos 12, 13 y 17 del Reglamento de Originalidad de Trabajos de Investigación, ***es procedente otorgar la constancia de originalidad*** para los fines que crea conveniente.

Ayacucho, 11 de junio del 2021

CCatamayo

Firmado digitalmente por Mg. Ing. Johnny Henrry Ccatamayo Barrios
Fecha: 2021.06.11 08:23:40 -05'00'

Mg. Ing. Ccatamayo Barrios Johnny Henrry
Verificador de originalidad de trabajos de tesis de pregrado de la FIMGC

Numero de constancia: 046-2021-FIMGC.

FACULTAD DE INGENIERÍA
DE MINAS, GEOLOGÍA Y CIVIL
Av. Independencia S/N
Ciudad Universitaria
Central Tel 066 312510
Anexo 151

MODELO DE GESTIÓN DE DESARROLLO DE SOFTWARE ÁGIL MEDIANTE SCRUM Y KANBAN SOBRE LA PROGRAMACIÓN EXTREMA, 2019

por Eder Solórzano Huallanca

Fecha de entrega: 10-jun-2021 08:18a.m. (UTC-0500)
Identificador de la entrega: 1604069632
Nombre del archivo: TESIS_SUSTENTADA_EDER_SOLORZANO_HUALLANCA.docx (2.67M)
Total de palabras: 27128
Total de caracteres: 144910

MODELO DE GESTIÓN DE DESARROLLO DE SOFTWARE ÁGIL MEDIANTE SCRUM Y KANBAN SOBRE LA PROGRAMACIÓN EXTREMA, 2019

INFORME DE ORIGINALIDAD

8%	6%	0%	5%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Universidad Nacional de San Cristóbal de Huamanga Trabajo del estudiante	4%
2	dspace.udla.edu.ec Fuente de Internet	1%
3	repositorio.unsch.edu.pe Fuente de Internet	1%
4	cdigital.uv.mx Fuente de Internet	<1%
5	www.researchgate.net Fuente de Internet	<1%
6	repositorio.unp.edu.pe Fuente de Internet	<1%
7	beagilemyfriend.com Fuente de Internet	<1%
8	ipmoguide.com Fuente de Internet	<1%

9	repositorio.utn.edu.ec Fuente de Internet	<1 %
10	arevalomaria.wordpress.com Fuente de Internet	<1 %
11	www.slideshare.net Fuente de Internet	<1 %
12	repositorioacademico.upc.edu.pe Fuente de Internet	<1 %
13	docplayer.es Fuente de Internet	<1 %

Excluir citas	Activo	Excluir coincidencias	< 30 words
Excluir bibliografía	Activo		