UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Modelo de desarrollo de software de la programación extrema sobre scrum para gestión de software ágil, 2019

Tesis para optar el título profesional de:

INGENIERO DE SISTEMAS

Presentado por:

Bach. Luis Miguel Prado Vasquez

Asesor:

Mg. Hubner Janampa Patilla

Ayacucho - Perú

2020

DEDICATORIA

Dedico el presente trabajo de investigación a mi estimada madre, quien me dio una de las herencias de mucho valor, mi profesión. Aún recuerdo esos momentos difíciles que afrontamos para salir adelante y fue el motivo me impulsó para seguir superándome cada día; a mi padre, quien estará orgulloso acompañándome cada día; a mis hermanos, quienes me motivaron para lograr mis objetivos profesionales; a mi asesor, por la orientación y apoyo que me brindó para la realización de mi trabajo de investigación.

AGRADECIMIENTO

Expreso mi sincero agradecimiento a los docentes de la Escuela Profesional de Ingeniería de Sistemas que han coadyuvado con sus enseñanzas y sabidurías a lo largo de mi formación profesional.

A mi madre, por el apoyo incondicional que me brindó en los momentos más difíciles.

A mi padre y a Dios, que en todo momento están conmigo ayudándome a lograr los objetivos propuesto.

CONTENIDO

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Los cambios tecnológicos constantes, constituyen un desafío para la industria del software e implica replantear los cimientos del proceso de construcción del software para responder las exigencias que demanda el mercado. En ese contexto, surgen las metodologías ágiles para el desarrollo de software. Estas metodologías ágiles comparten un punto en común que son los cuatro valores y doce principios del "Manifiesto Ágil", no obstante, difieren en sus orientaciones y propósitos para lograr sus objetivos, como es el caso de la metodología ágil de la Programación Extrema y el marco de trabajo Scrum; el primero está orientado a utilizar las buenas prácticas técnicas de desarrollo y pruebas; el segundo está orientado en las buenas prácticas de gestión y organización. En tanto, la Ingeniería de software considera que para obtener un producto software de calidad es necesario los aspectos técnicos de desarrollo y la administración del proyecto, por consiguiente, estas metodologías ágiles pueden complementarse entre sí para lograr un producto software con altos estándares de calidad.

El objetivo principal es la implementación del "Modelo de desarrollo de software de la Programación Extrema sobre Scrum para permitir la gestión de software ágil". El tipo de estudio es observacional, retrospectivo y transversal.

De acuerdo al capítulo III, en la que se logró la implementación del "Modelo de desarrollo de software de la Programación Extrema sobre Scrum" y posteriormente demostrado en el capítulo IV, sobre un caso práctico, obteniendo un producto software de altos estándares de calidad, en el tiempo establecido y escalable en el tiempo.

Palabras claves: Programación Extrema, Scrum, Gestión de software ágil, Ejecución del sprint, Metodologías ágiles, Integración continua, Jira.

INTRODUCCIÓN

La elección de una metodología ágil que incorpora las prácticas de gestión y organización, y al mismo tiempo que utilice las buenas prácticas de desarrollo, resalta una problemática muy común para la industria del software en la actualidad. Las metodologías poseen diferentes orientaciones y propósitos para lograr sus objetivos, a pesar de que mantienen en común los cuatro valores y los doce principios del "Manifiesto Ágil". En ese contexto, emergen las metodologías ágiles más utilizados: La Programación Extrema y el marco de trabajo Scrum; el primero está orientado a utilizar las buenas prácticas técnicos de desarrollo y pruebas; el segundo está enfocado en las buenas prácticas de organización y gestión. Como dice Kniberg (2015), "Scrum se centra en la gestión y prácticas de organización, mientras que XP se centra principalmente en las prácticas reales de desarrollo, abordan diferentes áreas y pueden complementan entre sí".

Mi motivación, para implementar un "Modelo de desarrollo de software de la Programación Extrema sobre Scrum", es proponer un modelo robusto que incorpora dos componentes fundamentales: las buenas prácticas de desarrollo y las prácticas de gestión y organización para obtener un producto software con altos estándares de calidad que busque la satisfacción del cliente.

En la actualidad, el uso de las metodologías ágiles de forma individual ya no tiene gran aceptación y respaldo como tuvieron en su momento, por diferentes situaciones, entre los cuales las exigencias del mercado cambiante que exigen metodologías que se enfocan en lo moderno y adaptativo que puedan combinarse unos con otros en base a los postulados de la Ingeniería de Software. Como dice Sommerville (2011), "la ingeniería de software no solo se interesa en los procesos técnicos del desarrollo del software, también lo hace con la administración del proyecto para obtener resultados de la calidad requerida dentro de una fecha establecida". En consecuencia, el surgimiento de las metodologías hibridas es una realidad, puesto que dos metodologías ágiles pueden complementan perfectamente, en base a un análisis comparativo de los eventos, prácticas, artefactos, etc. En esa línea planteamos un modelo robusto que incorpora los dos componentes

complementarios, argumentado por la Ingeniería de software para obtener un producto de calidad y que satisfaga la necesidad del cliente.

Los principales objetivos que se deben lograr son: a) Proponer un modelo de desarrollo de software iterativo e incremental de la Programación Extrema sobre Scrum que permita la gestión de software ágil a través de la planificación del Sprint. b) Proponer un modelo de desarrollo de software a través de las pruebas unitarias continuas de la Programación Extrema sobre Scrum que permita la gestión de software ágil a través de la ejecución del sprint. c) Proponer un modelo de desarrollo de software a través de la refactorización del código de la Programación Extrema sobre Scrum que permite la gestión de software ágil a través de la retrospectiva del Sprint.

CAPÍTULO I

PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1. DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

"Los principios ágiles son una de las mayores innovaciones en las metodologías de software. Uno de los puntos fuertes de la metodología ágil es que el trabajo se vuelve dinámico, aceptando los cambios que se puedan dar a lo largo del desarrollo, colaborando con los clientes y eligiendo al software por encima de la documentación exhaustiva" (Vlaanderen, Jansen, Brinkkemper y Jaspers, 2011). Estas características constituyen la esencia de todas las metodologías ágiles que surgieron en respuesta a los entornos cambiantes del mercado, proponiendo alternativas a las metodologías formales a las que consideraban rígidas y pesadas.

Actualmente existen muchas metodologías ágiles que se utilizan en la industria del software, pero las más utilizadas son: Programación Extrema, Scrum, Iconix, Kanban, todas ellas comparten un punto en común, el manifiesto ágil.

De acuerdo con los planteamientos de las características de cada metodología ágil, se ha logrado identificar que cada una de ellas posee diferentes orientaciones y propósitos, especialmente, la metodología ágil de la Programación Extrema y el marco de trabajo Scrum como sostiene Kniberg (2015), "Scrum se enfoca en la gestión y prácticas de organización, mientras que la Programación Extrema se centra más en las prácticas de desarrollo".

En la actualidad se necesita una metodología ágil que se caracteriza por contener ambos enfoques, tanto la gestión y organización como las prácticas de desarrollo para obtener un producto de alta calidad a nivel de desarrollo y que se pueda entregar en la fecha planificada. Teniendo en cuenta a Sommerville (2011), "la ingeniería de software no sólo se interesa en los procesos técnicos del desarrollo de software, también comprende la administración de proyectos de software donde se pueda obtener resultados de la calidad requerida dentro de una fecha establecida". En relación con este último, aseveramos que es factible combinar ambas metodologías ágiles de una forma positiva. Asimismo, en la opinión de Carrasco, Ocampo, Ulloa y Azcona (2019). "Las metodologías ágiles tuvieron gran acogida y respaldo en su momento, pero fueron reemplazados por nuevas

metodologías que se enfocan en lo moderno y adaptativo. El decir que estas metodologías pueden combinarse con otras es indiscutible y pueden complementarse mutuamente como Scrum y XP.

1.2. DEFINICIÓN DEL PROBLEMA DE INVESTIGACIÓN

PROBLEMA PRINCIPAL

¿Cómo el modelo de desarrollo de software de la Programación Extrema sobre Scrum permite la gestión de software ágil, 2019?

PROBLEMAS SECUNDARIOS

a. ¿Cómo el modelo de desarrollo de software iterativo e incremental de la Programación Extrema sobre Scrum permite la gestión de software ágil a través de la planificación del Sprint?
b. ¿Cómo el modelo de desarrollo de software a través de las pruebas unitarias continuas de la Programación Extrema sobre Scrum permite la gestión de software ágil a través de la ejecución del sprint?
c. ¿Cómo el modelo de desarrollo de software a través de la refactorización del código de la Programación Extrema sobre Scrum permite la gestión de software ágil a través de la retrospectiva del sprint?

1.3. OBJETIVOS DE INVESTIGACIÓN

OBJETIVO GENERAL

"Implementar el modelo de desarrollo de software de la Programación Extrema sobre Scrum para permitir la gestión de software ágil, 2019".

OBJETIVOS ESPECÍFICO

a. "Proponer un modelo de desarrollo de software iterativo e incremental de la Programación Extrema sobre Scrum que permita la gestión de software ágil a través de la planificación del Sprint".

b. "Proponer un modelo de desarrollo de software a través de las pruebas unitarias continuas de la Programación Extrema sobre Scrum que permita la gestión de software ágil a través de la ejecución del Sprint".

c. "Proponer un modelo de desarrollo de software a través de la refactorización del código de la Programación Extrema sobre Scrum que permite la gestión de software ágil a través de la retrospectiva del Sprint".

1.4. HIPÓTESIS DE LA INVESTIGACIÓN

No siempre las "investigaciones cuantitativas" proponen el planteamiento de la hipótesis. La formulación de la hipótesis está supeditado a un factor fundamental: El ámbito de aplicación inicial del estudio. "Los estudios de enfoque cuantitativo que enuncian hipótesis son aquellas cuyo propósito establece que su alcance es correlacional o explicativo, o las que poseen un alcance descriptivo que pretenden pronosticar una cifra o un hecho" (Hernández, Baptista y Fernández, 2014).

Supo (2012) refiere que "en las investigaciones que no llevan hipótesis no hace falta el planteamiento de hipótesis, por ende, no los hace menos importantes, sencillamente no llevan hipótesis, puesto que no es la intención del investigador, el propósito de la investigación no expresa la necesidad de aseverar o negar". En los estudios cuantitativos, que no tienen hipótesis generalmente indagan sobre la "cuantificación de la correlación entre las variables".

Siendo el estudio de nivel descriptivo, asimismo no se pretende pronosticar una cifra o un hecho y sobre la base de las consideraciones anteriores **no se realizará el planteamiento de hipótesis;** no obstante, lograremos el objetivo general y los objetivos específicos.

1.5. JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

1.5.1. IMPORTANCIA DEL TEMA

IMPORTANCIA TÉCNICA

"El Modelo de desarrollo de software de la Programación Extrema sobre Scrum" es una propuesta para el desarrollo de software que adopta dos componentes esenciales y complementarios: las mejores prácticas de desarrollo abordado por extreme programming y las prácticas de gestión y organización del "marco de trabajo Scrum". Esta propuesta permitirá a la comunidad de programadores y a la industria de software en general, construir un producto software con altos niveles de calidad tanto a nivel de desarrollo como en la administración del proyecto. Esta propuesta también garantiza la satisfacción del cliente al finalizar el proyecto.

IMPORTANCIA ECONÓMICA

La implementación del "Modelo de desarrollo de software de la Programación Extrema sobre Scrum", permitirá minimizar los costos y maximizar el retorno de inversión. La entrega de funcionalidades terminadas y de buena calidad en iteraciones cortas y planificadas ayudan a realizar solo las tareas necesarias, evitando trabajar en tareas que no agregan valor, minimiza los costos.

La obtención de un producto de mayor valor en menos tiempo de desarrollo producto de una adecuada planificación de las iteraciones donde participa el cliente, seleccionando los ítems del backlog del producto de mayor valor comercial y prioritarias maximiza el retorno de inversión.

1.5.2. JUSTIFICACIÓN

Existen diversas metodologías ágiles en la actualidad, pero las más utilizadas en la industria del software son la Programación Extrema y Scrum. De

acuerdo al planteamiento de las características de estas metodologías, se logra identificar que posee diferentes orientaciones y propósitos. Como afirma Kniberg (2015), “Scrum se centra en la gestión y organización, mientras que la programación extrema se centra en las prácticas reales de desarrollo”. Sin embargo, estas metodologías pueden complementarse entre sí. Como señalan Carrasco et al. (2019), “En la ingeniería de software existen decenas de metodologías que, en su momento, tuvieron gran acogida y respaldo, pero poco a poco fueron reemplazadas por nuevas metodologías que se enfocan en lo moderno y adaptativo. Estas metodologías pueden combinarse y complementarse mutuamente, como Scum y XP”. En esa misma línea Salazar et al. (2018) señalan que “es posible combinar las metodologías XP y Scrum de una manera efectiva, aplicando las prácticas y métodos definidos por XP a las reuniones y prácticas propuestas por Scrum”.

Por lo tanto, es indispensable la implementación del “Modelo de desarrollo de la Programación Extrema sobre Scrum”, que adopta las buenas prácticas de desarrollo de la “metodología ágil de la Programación Extrema” y las buenas prácticas de gestión y organización en el proceso de construcción del producto software propuesto por el “marco de trabajo Scrum” para obtener un producto software con valiosos estándares de calidad en la codificación y la entrega del producto dentro de los límites de tiempo y presupuesto pactado con el cliente.

.

1.5.3. DELIMITACIÓN DE LA INVESTIGACIÓN

La investigación estará enmarcada en el análisis de la “metodología de desarrollo de software ágil de la Programación Extrema” y el “marco de trabajo Scrum”, dado que la Programación Extrema emplea de forma correcta las prácticas técnicas de desarrollo y pruebas, mientras que Scrum se caracteriza por las buenas prácticas de gestión y organización de un proyecto software ágil.

CAPÍTULO II

REVISIÓN DE LA LITERATURA

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Según Mego (2020), en su trabajo de investigación titulado "Sistema de información web bajo la metodología XP y el marco de trabajo Scrum para la gestión académica del Instituto Superior Tecnológico Privado Ciro Alegría" de la Universidad Privada Unión de Perú, concluye que la aplicación efectiva en término de buenas prácticas de la metodología XP y el marco de trabajo Scrum brindan un soporte positivo en el planteamiento, planificación y elaboración del producto software.

Según Alfonzo, Mariño y Godoy (2011), en su trabajo de investigación titulado "Propuesta Metodológica para la Gestión de Proyecto de Software Ágil basado en la Web" de la Universidad Nacional de Nordeste de Argentina, afirman que Scrum es una metodología para la gestión y control de proyectos, centrada en la construcción de software que satisface las necesidades del cliente, cumple con los objetivos del negocio y el equipo de desarrollo que construye el producto software. Al no establecer prácticas de Ingeniería de Software, se combina con mucha facilidad con otras metodologías de desarrollo.

Según Mushtaq, Rizwan y Jameel (2012), en su proyecto de investigación denominado "Novel Hybrid Model: Integrating Scrum and XP", afirman que Scrum no proporciona ninguna dirección sobre cómo diseñar un producto software, mientras que el modelo de proceso XP está principalmente centrado en las prácticas de ingeniería en lugar de prácticas de gestión, por lo tanto es necesario la integración de las prácticas de gestión e ingeniería para construir un producto de calidad y valioso para los clientes.

2.2. MARCO TEÓRICO

2.2.1. MODELO DE DESARROLLO DE LA PROGRAMACIÓN EXTREMA SOBRE SCRUM

La Programación Extrema y Scrum se complementan perfectamente. Por parte de XP, se enfatizan en las prácticas, valores y el ciclo de vida, además está compuesto por seis fases. Respecto a Scrum, resaltan los eventos y artefactos. "La

combinación de estas dos metodologías significó una gran ayuda en el proceso de desarrollo de software, eludiendo la documentación exhaustiva y haciendo que el cliente sea un miembro más del equipo" (Carrasco et al.,2019).

"Scrum se enfoca en las prácticas de organización y gestión, mientras que XP se centra más en las prácticas de desarrollo. Esa es la razón de que funcionan tan bien juntas: Tratan de áreas diferentes y se complementan entre ellas" (Kniberg, 2015).

Salazar, Tovar, Linares, Lozano y Valbuena (2018) sostienen que cada metodología tiene distintas orientaciones y propósitos; en particular las siguientes metodologías: Scrum y XP, el primero enfatiza en los aspectos de gestión y organización de un proyecto software, mientras que el segundo maneja adecuadamente los aspectos técnicos de desarrollo. Por consiguiente, se pueden combinarse y complementarse de una forma adecuada, puesto que es factible realizar las actividades propuestas por la metodología Scrum, empleando las prácticas expuestos por la programación extrema.

A. SOFTWARE ITERATIVO E INCREMENTAL

el software se desarrolla en varias iteraciones, cada iteración abarca desde la planificación hasta la entrega. En cada iteración parte del sistema se desarrolla, prueba y mejora mientras se desarrolla una nueva parte. En cada iteración, se mejorará la funcionalidad. Además, el sistema está creciendo de manera incremental a medida que se agregan nuevas funciones en cada versión. Después de cada iteración, se entregará una versión al cliente para recibir comentarios (Abrahamsson, Conboy, Morgan, Baskerville, Fitzgerald y Wang, 2008).

El enfoque iterativo permite entregar un producto software de gran tamaño a partir de piezas o incrementos más pequeños. Los enfoques iterativos se utilizan con frecuencia en proyectos de desarrollo de software para promover la velocidad y la adaptabilidad, ya que el beneficio de la iteración es que puede ajustarse a medida que avanza en lugar de seguir una ruta lineal. Uno de los objetivos de un enfoque ágil o iterativo es liberar beneficios durante todo el proceso y no solo al final (Vieyra, 2020).

B. PRUEBAS UNITARIAS CONTINUAS

Como plantean Feathers y Martin (2014), una prueba unitaria es un fragmento de código que invoca una unidad de trabajo y verifica un resultado final especifico de esa unidad de trabajo. Si las suposiciones sobre el resultado final resultan ser incorrectas, la prueba unitaria ha fallado. Una prueba unitaria puede abarcar tan poco como un método o tanto como varias clases.

Bahit (2012) refiere que, “son test que se encargan de verificar de manera simple y rápida el comportamiento de una parte mínima de código de forma independiente y sin alterar el funcionamiento de otras partes de la aplicación”.

Citando a Khorikov (2020), “Una prueba unitaria es una prueba automatizada que verifica un pequeño fragmento de código (también conocido como unidad), lo hace rápido y de forma aislada al resto de la aplicación”.

C. REFACTORIZACIÓN DEL CÓDIGO

“Un cambio realizado en la estructura interna del software para que sea más fácil de entender y más barato de modificar sin cambiar su comportamiento observable” (Fowler, 2019).

La refactorización es el medio principal de XP para alentar a los desarrolladores a realizar cambios en el código que ya está funcionando. Dar la bienvenida al cambio (además de la práctica de "Diseño simple" de XP), puede resultar en las suposiciones estructurales originales de una clase o método (o incluso un subsistema completo) muy alejado de lo que finalmente termina siendo. Refactorizar es una cuestión de traer toda la información que se ha aprendido desde que se escribió el código para tener en cuenta y reelaborarlo para adaptarlo mejor a su uso. (Koch, 2005).

2.2.2. GESTIÓN DE SOFTWARE ÁGIL

Sommerville (2011) postula que la gestión de proyectos de software es una parte fundamental de la Ingeniería de Software. La gestión de proyecto no afianza el éxito de un proyecto software, no obstante, una inadecuada gestión conduce a la ruina. El producto es entregado tardíamente, los costes son excesivos que los planificados al inicio y los requisitos no se concretan en la fecha prevista, por consiguiente, una gestión adecuada de un proyecto software está sujeto a la

planificación de su progreso, porque la ingeniería de software siempre está supeditada a limitaciones con respecto al tiempo y presupuesto.

Fitsilis (2008) sostiene que, “la gestión de proyectos ágil de software se basa en los siguientes principios: abrazar el cambio, centrarse en el valor para el cliente, entregar parte de la funcionalidad de forma incremental”.

A. PLANIFICACIÓN DEL SPRINT

La reunión de planificación del sprint permite al equipo organizar su trabajo y comprometerse con el objetivo de sprint, sentando así las bases para su autoorganización. El propietario del producto tiene la responsabilidad de asegurarse de que la pila del producto esté bien preparada, sus ítems priorizados y sus elementos de alta prioridad detallados, antes de la reunión de planificación del sprint. También se espera que asista a esta reunión para aclarar requisitos y responder interrogantes, su función durante la planificación del sprint es ayudar al equipo a comprender qué se debe hacer. El equipo de desarrollo realiza la estimación para determinar los ítems que se pueden entregar al finalizar el sprint. El propietario del producto no está autorizado a decirle al equipo cómo se debe hacer el trabajo durante el sprint o para identificar tareas en nombre del equipo, son responsabilidad exclusiva del equipo. El equipo debe comprometerse a realizar el trabajo de manera realista en base a su capacidad, limitar la cantidad de trabajo por sprint a la capacidad del equipo crea un ritmo sostenible (Pichler, 2010).

la planificación del sprint se realiza al inicio del sprint. Durante esta reunión, el equipo y el propietario del producto acuerdan la meta del sprint. Luego, el equipo selecciona un subconjunto de alta prioridad de los ítems del backlog del producto que se pueden completar durante un sprint, asumiendo que el equipo trabaja a un ritmo sostenible. Para adquirir confianza en lo que se puede hacer, muchos equipos de desarrollo desglosan cada característica específica en un conjunto de tareas. La recopilación de estas tareas, junto con sus items de la pila del producto asociados, forma una segunda pila, llamado Sprint backlog (Rubin, 2013).

B. EJECUCIÓN DEL SPRINT

"La ejecución del sprint es el trabajo necesario que efectúa el equipo Scrum para alcanzar los objetivos del sprint. Comienza después de la planificación del sprint y finaliza cuando comienza la revisión del Sprint" (Rubin, 2013).

C. RETROSPECTIVA DEL SPRINT

La retrospectiva del sprint ocurre al final de cada sprint y es el momento de inspeccionar y adaptar el proceso. Es el espíritu de la mejora continua, el Scrum Master, el propietario del producto y el equipo de desarrollo se unen para discutir lo que ha funcionado y lo que no funcionado con Scrum y prácticas técnicas asociadas. 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 serán implementados en el próximo sprint (Rubin, 2013).

"Es una reunión de lecciones aprendidas realizada al final de cada Sprint. La reunión se basa en el Sprint anterior y se discuten las oportunidades de mejora de los sprints futuros" (Canty, 2015).

2.3. MÉTODOS Y TÉCNICAS

2.3.1. LA INGENIERÍA DE SOFTWARE

Es una disciplina que abarca todo el proceso de la creación de software, empezando desde las etapas más tempranas hasta la etapa de mantenimiento posterior a la implementación del software. "La ingeniería de software se interesa tanto por los procesos técnicos del desarrollo del software, así como de la administración del proyecto empleando teorías y métodos para alcanzar resultados de calidad dentro de una fecha señalada" (Sommerville, 2011).

La ingeniería de software emplea un conjunto de mecanismos y técnicas que tienen como objetivo crear un producto software computacional de forma metódica y disciplinada. La Ingeniería de Software se distingue de la programación ordinaria por el foco riguroso de la calidad del producto final y de la automatización de los procesos que se utilizan para la creación y el mantenimiento posterior de un producto software con altos estándares de calidad (Arias, 2015).

2.3.2. EL MANIFIESTO ÁGIL

En marzo de 2001, 17 críticos de los modelos de producción basados en procesos, convocados por Kent Beck, que había publicado un par de años antes el libro en el que explicaba la nueva metodología Extreme Programming (Beck, 2000) se reunieron para discutir sobre el desarrollo de software. En la reunión se acuñó el término "Métodos Ágiles" para definir a aquellos que estaban surgiendo como alternativa a las metodologías formales a las que consideraban excesivamente "pesadas" y "rígidas" por su carácter normativo y fuerte dependencia de planificaciones detalladas, previas al desarrollo.

Los integrantes de la reunión resumieron en cuatro postulados lo que ha quedado denominado como "Manifiesto Ágil", que son los valores sobre los que se asientan estos métodos.

2.3.3. VALORES DEL MANIFIESTO ÁGIL

El manifiesto por el desarrollo de software ágil dice: "Estamos poniendo al descubierto mejores métodos para desarrollar software, haciéndolo y ayudando a otros a que lo hagan. Con este trabajo hemos llegado a valorar":

- **"Individuos e interacciones"**, sobre procesos y herramientas
- **"El software que funciona"**, sobre documentación completa.
- **"Colaboración con el cliente"**, sobre la negociación de contratos.
- **"La respuesta al cambio"**, sobre seguimiento de un plan.

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

2.3.4. PRINCIPIOS DEL MANIFIESTO ÁGIL

El manifiesto ágil, después de los postulados de los cuatro valores en los que se fundamenta, establece estos doce principios:

1. Nuestra máxima prioridad es la satisfacción del cliente mediante la entrega temprano y constante de producto software de alto valor.
2. Bienvenidos los requisitos cambiantes, incluso al final del desarrollo. A través de los procesos ágiles aprovechamos el cambio para que el cliente obtenga una ventaja competitiva.

3. Entregue producto que funcione de forma periódica, desde un par de semanas hasta un par de meses, preferentemente a la escala de tiempo más corta.
4. Las personas de negocios y los desarrolladores deben trabajar juntos, a diario y durante todo el proyecto.
5. Construya proyectos en torno a personas motivadas. Debe darse a estos el ambiente y el apoyo que necesiten, y confiar en que harán el trabajo.
6. El método más eficiente y eficaz para transmitir información a los integrantes de un equipo de desarrollo, y entre estos, es la conversación cara a cara.
7. El software que funciona es la principal medida de progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben poder mantener un ritmo constante en forma indefinida.
9. La atención continua a la excelencia técnica y el buen diseño mejora la agilidad.
10. Es esencial la simplicidad: el arte de maximizar la cantidad de trabajo no realizado.
11. Las mejores arquitecturas, requisitos y diseños surgen de la autoorganización de los equipos.
12. Reflexiona y realice ajustes periódicamente sobre tu forma de trabajar para aumentar la eficacia.

2.3.5. AGILE

"La agilidad es un comportamiento o habilidad persistente de una entidad que exhibe flexibilidad para adaptarse a lo esperado o inesperado, cambia rápidamente; La agilidad se puede evaluar por la flexibilidad, la velocidad, aprendizaje y capacidad de respuesta" (Qumer y Henderson-Sellers, 2008).

La agilidad se define en términos de la adopción de los métodos ágiles. La agilidad es la disposición continua para crear cambios rápidos de forma inherente, adoptar cambios de forma proactiva o reactiva y aprender del cambio mientras contribuye a la percepción de valor al cliente (Conboy, 2009).

2.3.6. PROGRAMACIÓN EXTREMA (XP)

Es una metodología de alta disciplina que requiere el cumplimiento de los estándares y un fuerte compromiso con las pruebas unitarias, refactorización e integración. La falta de enfoque en la salida de documentos que caracteriza a los equipos XP es una expresión agresiva del "software de trabajo en lugar de documentación" en manifiesto ágil. De hecho, XP evita la mayoría de las formas de gasto administrativos y se centra completamente en las prácticas técnicas de ingeniería, XP aboga por los generalistas que pueden contribuir a muchas facetas de un proyecto, trabajando en estrecha colaboración y compartiendo sus conocimientos (Moran, 2015).

Es una colección de prácticas de ingeniería de software conocidas. XP tiene como objetivo obtener un producto software de calidad a pesar de los constantes cambios de requisito. La novedad de XP se basa en la forma en que se recopilan y alinean las prácticas individuales para que funcionen entre sí. Algunas de las principales características son las iteraciones cortas, realimentación rápida, participación del cliente, comunicación y coordinación constante, refactorización continua, integración y pruebas continuas, propiedad colectiva del código y programación en pares (Dingsoyr, Dyba y Moe, 2010).

A. VALORES DE LA PROGRAMACIÓN EXTREMA

Beck (1999) plantea: "XP es una filosofía de desarrollo de software basado en valores de la comunicación, retroalimentación, sencillez, el coraje y el respeto".

a. LA COMUNICACIÓN

"La comunicación constante es fundamental en XP, el dialogo frontal, cara a cara, entre desarrolladores, administrador y el cliente es el medio básico de comunicación. Una buena comunicación se debe mantener durante todo el proyecto" (Beck, 1999).

Para desarrollar software de calidad de manera eficiente, clara y frecuente. Se requiere comunicación entre el cliente y los miembros del equipo. Sentarse con el cliente y sacar ideas, hacer que la comunicación sea muy interactiva y no pasarse

escribiendo un documento de varias paginas intentando obtener lo mismo (Blankenship et al., 2011).

b. LA SIMPLICIDAD

"XP apuesta por la sencillez en su máxima expresión. Sencillez en el diseño, en código, en los procesos, etc. La sencillez es imprescindible para que todos entiendan el código y se trata de mejorar a través re codificaciones continuas" (Beck, 1999).

Se pretende desarrollar solo lo necesario y no dilapidar el tiempo en detalles que no sean requeridos en el momento. En este aspecto, se asemeja a otra metodología ágil, denominada Kanban, en la cual, un proceso "anterior" solo produce lo que el proceso posterior demanda (Bahit, 2012).

c. LA REALIMENTACIÓN

El cliente es entrevistado utilizando ciclos de retroalimentación muy cortos. El progreso del desarrollo se muestra con frecuencia y, de este modo, se pueden evitar errores. La realimentación también proviene de las pruebas ejecutadas (Parsons y MacCallum, 2019).

"El objetivo de la Programación Extrema es entregar lo necesario al cliente, en el menor tiempo posible. A cambio, demanda al cliente, una realimentación a fin de conocer sus requerimientos e implementar los cambios tan pronto como sea posible" (Bahit, 2012).

d. EL CORAJE

Al identificar problemas de consideración en el diseño, o en alguna fase del ciclo de la programación extrema, debemos tener mucho coraje para hallar la solución, sin importar el grado de dificultad que demanda. Si es necesario debemos sustituir totalmente parte del código escrito, sin importar el tiempo de desarrollo que se ha empleado para realizar el trabajo (Beck, 1999).

Usar los valores de XP requiere mucho coraje. Es importante contar la verdad sobre el progreso y las estimaciones. Además, XP descarta la palabra al

momento de realizar la refactorización del código antiguo o desechar para realizar los cambios (Parsons y MacCallum, 2019).

e. EL RESPETO

Todo el mundo da y recibe respeto en el equipo. Todos se valoran como miembros del equipo. El equipo de desarrollo respeta el conocimiento del cliente y viceversa. El equipo directivo respeta los derechos del equipo a aceptar la responsabilidad y a tener autoridad sobre su propio trabajo (Canty, 2015).

"Tratarnos a nosotros mismos y a los demás con dignidad y reconocer la experiencia y nuestro mutuo deseo de éxito" (Shore y Warden, 2008).

B. LAS PRÁCTICAS DE LA PROGRAMACIÓN EXTREMA

Blankenship et al. (2011) manifiesta que XP está compuesto por un conjunto de doce prácticas y son los siguientes:

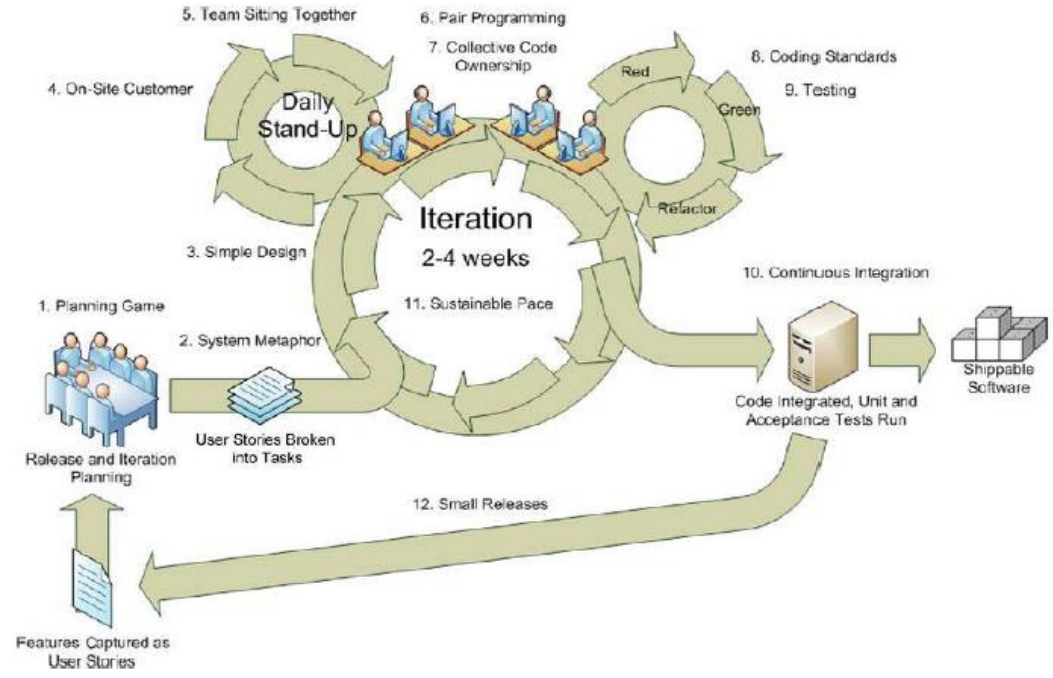


Figura 1. Las prácticas de XP

Fuente: adaptado de Blankenship et al. (2011)

a) El juego de la planificación (Planning game)

El cliente y el equipo de desarrollo deben tomar una decisión de las historias de usuario que se incluirán en la próxima iteración, la prioridad de cada

historia y cuándo se debe realizar el lanzamiento. Los desarrolladores son responsables para dividir las historias en un conjunto de tareas y estimar la duración de cada tarea. La suma de las duraciones le dice al equipo lo que realmente creen que pueden hacer antes de la fecha de entrega del lanzamiento. Tenga en cuenta que la estimación es responsabilidad de los desarrolladores y no del cliente o el gerente. En XP, solo los desarrolladores hacen estimaciones (Dooley, 2017).

b) La metáfora del sistema (Metaphor)

"La idea general del proyecto; el objetivo amplio contado como una narración o historia, para mantener la jerga técnica al mínimo y construir una visión colaborativa entre desarrolladores y clientes" (Freedman, 2016).

"XP usa metáforas para llegar a un entendimiento común de diseños y establecer una visión técnica compartida" (Canty, 2015).

c) El diseño simple (Simple design)

XP admite un diseño simple en el que el desarrollador puede realizar cambios rápida y fácilmente. El diseño debe permanecer simple porque solo las cosas más simples que funcionan se incluyen sin estructuras o características complicadas. La complejidad del diseño aumenta riesgo e históricamente ha resultado en proyectos fallidos (Canty, 2015).

Pressman (2012) señala que, siempre se prefiere un diseño simple sobre sobre una representación más compleja, porque el diseño guía la implementación de una historia conforme se escribe: "nada más y nada menos". Se desalienta el diseño de funcionalidad adicional, porque el desarrollador infiere que se requerirá después.

d) Cliente en sitio

"XP pide que el cliente sea completamente integrado al equipo de desarrollo, disponible para revisar características, compilaciones y pruebas, y para revisar, evaluar y optimizar el producto a medida que evoluciona" (Freedman, 2016).

durante una iteración, es ideal que el cliente esté en el sitio para ayudar a responder rápidamente los detalles de una historia de usuario. El punto importante es tener un cliente disponible que pueda proporcionar respuestas rápidamente a los desarrolladores para explorar los detalles de una historia de usuario (Blankenship et al., 2011).

e) La programación en pareja (Pair programming).

el proceso mediante el cual dos miembros del equipo colaboran, discuten, codifican y prueban una pieza de funcionalidad, es una de las prácticas más interesantes de XP. La programación en pareja proporciona revisión inmediata del código y su caso de prueba previsto, y es una de las principales razones por las que XP induce la calidad de código en el producto (Kruchten, 2007).

Koch (2005) manifiesta que la práctica de XP de "Programación por pares" es única y ajena a casi todas las organizaciones. Estos desarrolladores trabajan juntos durante todo el trabajo técnico como diseño, desarrollo de casos de prueba, codificación y prueba. Mientras un miembro de una pareja está "manejando" la computadora, el otro observa y evalúa el trabajo del socio haciendo preguntas. La pareja cambia de roles en forma regular. Aunque esta práctica suena como un desperdicio de recursos, los defensores de XP afirman todo lo contrario por una variedad de razones:

- La revisión se realiza en tiempo real, puesto que el socio observador está pendiente de los posibles errores en caso que se produzca se realiza la corrección inmediatamente.
- La interacción constante entre los individuos, permite aprender unos de otros.
- Permite que los desarrolladores se familiaricen con cada parte de sistema.
- Proporciona una forma para que cualquier miembro nuevo del proyecto pueda convertirse en un miembro contribuyente del proyecto muy rápidamente.

f) La propiedad colectiva del código.

"Los programadores deben aceptar la idea de que cualquier integrante del proyecto puede cambiar su código y que la propiedad colectiva se extiende a todo el proyecto; esto es un proyecto de equipo, no uno individual" (Dooley, 2017).

La propiedad colectiva es una práctica importante para el equipo, porque proporciona una comprensión clara de los roles y responsabilidades de todos los miembros del equipo con respecto al software. Se ha descubierto que esta práctica mejora la calidad de software producido por equipos de proyecto. Dando a todos los miembros del equipo una responsabilidad compartida para el software, la propiedad colectiva fomenta las prácticas de mejora de calidad y la refactorización (Smite, Moe y Agerfalk, 2010).

g) Los estándares de codificación.

Schmidt (2016) sostiene que, es un conjunto de reglas y convenciones de los desarrolladores de software, equipo de desarrollo o comunidad. Incluye un estilo de programación común que mejoran la legibilidad y la capacidad de mantenimiento de un código de software.

"Manejar estándares y mejores prácticas al escribir códigos ayuda a evitar dialectos particulares de cada desarrollador, y por tanto la comprensión del código será óptima" (Salazar et al., 2018).

"Aquí se ha especificado un estilo y formato para el código fuente. Esta práctica mantiene el código consistente, facilitando la lectura, interpretación y refactorización que se pueda dar en un futuro" (Carrasco et al., 2019).

h) Test Driven Development (Desarrollo impulsado por pruebas)

Es una práctica de desarrollo iterativa en la que los desarrolladores primero escriben un caso de prueba para la funcionalidad del software deseado, para verificar si el producto software incluye la funcionalidad deseada. Solo entonces, los desarrolladores escriben código para pasar ese caso de prueba (Schmidt, 2016).

Gulati y Sharma (2017) manifiestan que, es una práctica de desarrollo que aumenta la confianza del desarrollador al promover pruebas para todos los requisitos del software. Nos hace trabajar en ciclos de desarrollo incrementales

cortos, proporcionando así una rápida retroalimentación sobre nuestro progreso. TDD nos obliga a escribir una prueba fallida antes escribir el código de producción.

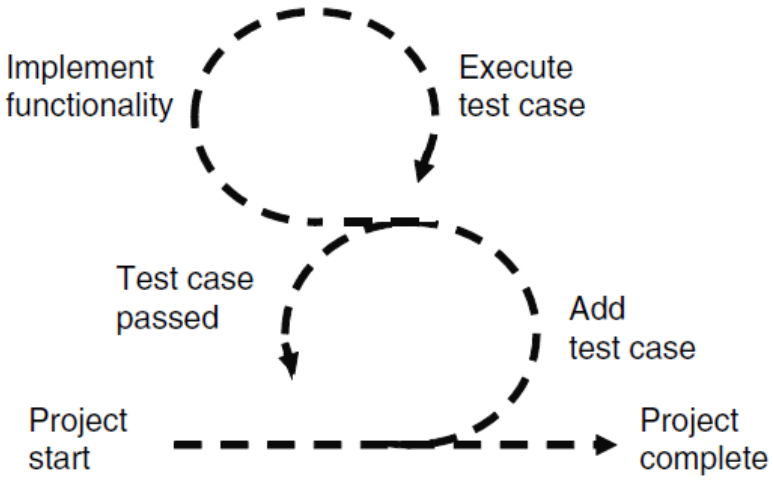


Figura 2. Ciclo de desarrollo guiado por pruebas

Fuente: Adaptado de Stober y Hansmann (2010)

Como señala Jurado (2010), "el desarrollo dirigido por pruebas es una técnica de diseño e implementación de software incluida dentro de la metodología XP". El desarrollo dirigido por pruebas se agrupa en tres pilares fundamentales:

- Solamente se debe implementar las funciones que el cliente requiere.
- Minimizar los defectos que suelen llegar a la fase de producción.
- Implementación de software modular, cuya característica es la reutilización y la preparación para el cambio.

Esta práctica de XP, consta de tres tipos de pruebas y son las siguientes: Las pruebas unitarias, pruebas de aceptación y las pruebas de integración

1. Las pruebas unitarias

"consisten en probar los métodos de manera individual mientras se va programando. Técnica conocida como Unit Testing, la cual forma parte de la técnica TDD (Test driven development – desarrollo guiado por pruebas)" (Bahit, 2012).

Según Gulati y Sharma (2017), Todas las pruebas unitarias deben tener las siguientes características:

Legible. Una prueba unitaria debe tener un nombre significativo para comprender el comportamiento de la unidad que se está probando sin mirar los detalles de la implementación.

Rápido. las pruebas deben ejecutarse en pocos segundos para que proporcionen realimentación. Si las pruebas toman más tiempo, el programador buscará formas de omitir las pruebas.

Independiente y aislada. las buenas pruebas unitarias son independientes del orden de ejecución. No dependen de otras pruebas unitarias para que funcione correctamente.

Correcto. una buena prueba unitaria hace lo que dice. Un caso de prueba debe corresponden a un solo caso, es decir, comportamiento.

Independiente del medio ambiente. La mayoría de las veces, encontramos que las pruebas unitarias fallan porque dependen de algún factor externo. El factor externo podría ser un archivo en una ubicación particular, una variable de entorno o algo más. Esto conduce a pruebas frágiles. Una buena prueba unitaria no depende del medio ambiente.

Repetible. una buena prueba unitaria produce el mismo resultado cada vez que se ejecuta. La ejecución de la prueba debe automatizarse utilizando herramientas de compilación.

2. Las pruebas de aceptación

pruebas de aceptación son indicadores de la finalización de un requisito o característica. Cuando se aprueban todas las pruebas de aceptación de un requisito o función, se dice que está hecho (Koskela, 2008).

“Están más orientados a las pruebas de funcionalidad, es decir al comportamiento funcional del código. Estas pruebas, a diferencia de los Test Unitarios, son definidas por el cliente” (Bahit, 2012).

i) La integración continua (Continuous integration)

Este principio significa que el código para cada historia está integrado en el sistema en evolución tan pronto como como esté listo. Cuando los desarrolladores terminan de escribir una historia la integran inmediatamente. Con varios desarrolladores trabajando simultáneamente en historias, esto da como resultado que la integración ocurre casi todo el tiempo. El valor de esta práctica es que proporciona la realimentación lo más rápido posible a los desarrolladores que acaban de implementar las historias de usuario (Koch, 2005).

Es una práctica de desarrollo de software donde cada desarrollador trabajar en una base de código en particular e integra continuamente los nuevos desarrollos o código de software modificado para evitar problemas de integración al finalizar el proyecto. Para realizar el proceso de integración se utilizan herramientas automatizadas (Schmidt, 2016).

j) La refactorización.

Es el proceso de mejorar y simplificar el diseño del código existente; manteniendo la misma funcionalidad, pero produciendo un mejor diseño (porque es más simple). La refactorización permite automatizar pruebas para ser escritas y hace que la aplicación sea más fácil de mantener (Blankenship et al., 2011).

Freedman (2016) manifiesta que la refactorización es la optimización del código interno y arquitectura de software, y es un elemento clave de XP. También es una respuesta a uno de los peligros de diseño iterativo, el peligro de que las iteraciones separadas estará mal integrado e internamente incompatible. La refactorización es un enfoque disciplinado para reconstruir el sistema interno para asegurar simplicidad, elegancia y compatibilidad.

k) El ritmo sostenible

"XP entiende que el equipo debe mantener un nivel constante de productividad. Esto solo se puede lograr si el equipo mantiene un ritmo sostenible que se traduce en 40 horas semanales de trabajo" (Canty, 2015).

Desarrollar software para cumplir con la fecha límite del cliente es, sin duda, exigente para los miembros del equipo del proyecto. El proceso de desarrollo

de software es, por naturaleza, una tarea compleja e intensiva en conocimientos. Los miembros del equipo a menudo registran una gran cantidad de horas, trabajando hasta tarde para producir software funcional dentro de un plazo establecido. Tales demandas inevitablemente crean estrés y agotamiento en los miembros del equipo. Además, aumenta la probabilidad de que se introduzcan o se produzcan defectos. El principio de ritmo sostenible asegura que los miembros del equipo trabajen en un horario cómodo que se puede mantener durante la duración del proyecto. Con un ritmo sostenible, los desarrolladores pueden aportar más energía a su trabajo en el proyecto, porque no tienen que trabajar durante una cantidad excesiva de horas (Smite et al., 2010).

l) Pequeños lanzamientos

La práctica de implementar pequeñas versiones de software funcional refleja un proceso para entregar el producto al cliente. Este enfoque permite a los equipos de proyecto centrarse primero en entregar la funcionalidad más crítica al cliente. Luego, se pueden agregar de forma iterativa otros componentes importantes del software en versiones posteriores. Este enfoque también proporciona flexibilidad para incorporar comentarios y sugerencias de los clientes e incluir en la próxima versión. Este enfoque iterativo para entregar el software funcional a menudo produce una alta satisfacción al cliente (Smite et al., 2010).

XP promueve lanzamientos frecuentes, que pueden ser relativamente pequeños, pero destaca las funciones priorizadas por el cliente. XP no permite que el equipo de desarrollo se esconda durante un tiempo prolongado, esperando que el proyecto se complete a tiempo. Esta transparencia de los lanzamientos frecuentes anima a los clientes al mostrarles que el equipo está agregando valor al proyecto a lo largo del camino (Blankenship et al., 2011)

C. FASES DE LA PROGRAMACIÓN EXTREMA

Según Letelier y Penadés (2006), “el ciclo de vida de XP consta de seis fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y muerte del proyecto”.

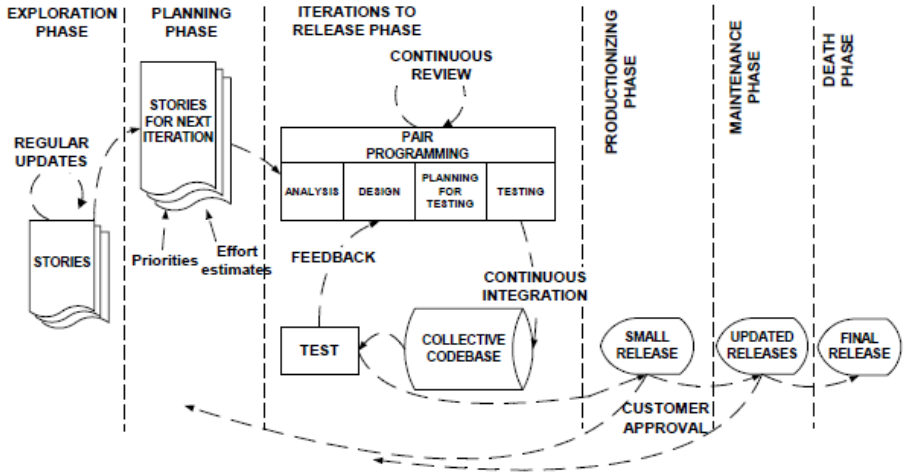


Figura 3. Fases de XP

Fuente: Adaptado de Beck y Andrés (2005)

a. Fase de exploración. Esta fase comprende dos etapas: Modelado de requisito iniciales y Modelado de la arquitectura inicial.

- **Modelado de requisitos iniciales.** En esta etapa se reúnen los clientes y usuarios con el equipo de desarrollo para crear las "historias de usuario". Las historias de usuario son extremadamente relevantes en XP, puesto que proporcionan requisitos de alto nivel para su sistema y son una entrada fundamental en el proceso de planificación.
- **Modelado de la arquitectura inicial.** Esta etapa consiste en la identificación de una metáfora que describa cómo piensa construir el sistema. La metáfora actúa como un marco conceptual, identificando objetos clave y brindando información sobre sus interfaces. No importa cuan exactamente se desarrolla esta metáfora una vez que esta al comienzo del proyecto.

b. Fase de planificación de la entrega. Durante esta fase los desarrolladores se reúnen con el cliente para implementar el primer conjunto de historias de usuario de mayor prioridad en la versión actual. Normalmente se elabora una tarjeta de tarea, que es una lista escrita de lo que se debe hacer para completar la tarea, también se pueden elabora bocetos y luego se transfieren a una tarjeta de índice para tener un registro de su diseño en el que se plasma la estimación del esfuerzo

de codificación durante el próximo sprint. También se establece el cronograma y presupuesto de este lanzamiento.

c. Fase de iteraciones para lanzamiento. En esta fase se realizan los esfuerzos de desarrollo, incluido el modelado, la programación, las pruebas y la integración. En la planificación de iteraciones, la atención se centra en las historias de usuario asignadas a la iteración actual, encontrará que se han agregado nuevas historias de usuario, las que no fueron estimados, por lo que debe realizar el esfuerzo de identificar las tareas para poder estimar con precisión cada historia de usuario. Después de estimar las nuevas historias, es posible que descubra que tiene demasiadas o muy pocas historias para la iteración y necesitará mover hacia o desde otras iteraciones.

d. Fase de producción. Consiste en asegurar que las pruebas adicionales de funcionalidad y rendimiento están hechas (prueba del sistema, prueba de carga, instalación pruebas). La determinación de los nuevos cambios necesarios incluido en la versión actual. Implementar y probar los nuevos cambios. Documentar las ideas pospuestas y sugerencias para implementarlas durante fase de mantenimiento o en próximas versiones. Entrega de la versión actual en ejecución a los clientes.

e. Fase de mantenimiento. esta fase es para asegurar que el sistema se esté utilizando correctamente y que funcione bien. Aquí se debe hacer un esfuerzo para apoyar al cliente, es vital que se mantenga un sistema y para hacer esto, los usuarios deben saber lo que están haciendo; es por eso que la documentación y también la participación del usuario durante el desarrollo es crucial. sí un sistema no se mantiene, su desarrollo es una pérdida de tiempo.

f. Fase de muerte del proyecto. Esto ocurre cuando los clientes no tienen más historias para implementar, se satisfacen las necesidades del cliente. toda la documentación necesaria se escribe en esta fase. esta fase también puede ocurrir si un desarrollo tiene que abandonarse debido a que es demasiado caro para un desarrollo posterior o si no se está entregando el producto deseado.

D. ROLES DE LA PROGRAMACIÓN EXTREMA (XP)

A juicio de Beck y Andrés (2005), los roles de XP son:

Programador. La responsabilidad del desarrollador de un proyecto XP es crear una funcionalidad basada en las historias de usuario proporcionadas por el cliente. Este rol debe comprender e implementar las características del producto trabajando muy de cerca con el cliente. Las tareas son creadas a partir de historias de usuario y las estimaciones correctas, permiten al cliente decidir cuál de las historias tienen mayor prioridad para implementar en la cada una de las iteraciones. Los desarrolladores son los empoderados para estimar las tareas sin ninguna injerencia del cliente.

Cliente. Es el que define los requisitos para el proyecto y establece el objetivo del proyecto. Asimismo, el cliente toma las decisiones comerciales más difíciles y trabaja muy de cerca con los desarrolladores de software. Cuanto más involucrado esté el cliente, mayor será la probabilidad de éxito en un proyecto XP. En realidad, el cliente es la voz del usuario final del producto. Este rol debe aclarar las características del producto y escribe las historias de usuario para el equipo. Finalmente, este rol ejecuta las pruebas funcionales para validad la implementación.

Encargado de pruebas (Tester). Es el encargado de ejecutar las pruebas regularmente y difundir los resultados dentro del equipo, además es el responsable de las herramientas de soporte para pruebas.

Encargado de seguimiento (Tracker). Este rol es complementario y no existe en todos los proyectos XP. El Tracker es el responsable de mantener el cronograma del proyecto. Este rol también mide y rastrea la velocidad del equipo. Hace un seguimiento regular del progreso del equipo y realiza ajustes que garanticen que las iteraciones se mantienen con el objetivo.

Entrenador (Coach). El entrenador es otro rol complementario que puede no existir en todos los proyectos XP. Este rol es responsable de brindar

orientación y tutoría al equipo. El rol del entrenador es asegurar que el equipo comprenda las prácticas de XP en conjunto con el desarrollo de software. Este rol también ayuda con la resolución de problemas y funciona como árbitro entre el cliente y el equipo de desarrollo cuando sea necesario.

E. LOS ARTEFACTOS DE XP

HISTORIA DE USUARIO

Son descripciones simples y cortas que describen las funcionalidades que deberán ser implementadas. Deben ser escritas por el propio cliente, empleando sus propias palabras, y generalmente son registradas en tarjetas. Las historias de usuario contienen una breve descripción que representa una necesidad del cliente (Laínez, 2015). A continuación, se muestra el esquema de una plantilla comúnmente utilizado.

Tabla 1
Plantilla para las historias de usuario

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

Fuente: Anónimo

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).

Tabla 2
Plantilla para tareas de ingeniería

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:	

Fuente: Anónimo

PRUEBAS DE ACEPTACIÓN

"Se centran en las características y funcionalidad generales del sistema que son visibles y revisables por parte del cliente. Las pruebas de aceptación se derivan de las historias de los usuarios que se han implementado como parte de la liberación del software" (Pressman, 2010).

Tabla 3
Plantilla de las pruebas de aceptación

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

Fuente: Anónimo

TARJETAS CRC (CLASE-RESPONSABILIDAD-COLABORACIÓN)

"Es un conjunto de tarjetas índice estándar que representan clases. Se dividen en tres secciones. En la parte superior se escribe el nombre de las clases, en la parte izquierda se enlistan las responsabilidades de la clase y en la derecha, los colaboradores" (Pressman, 2010).

Tabla 4

Plantilla de las tarjetas CRC

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.

Fuente: Anónimo

2.3.7. MARCO DE TRABAJO SCRUM

Sommerville (2011) afirma: "Scrum es un método ágil que ofrece un marco de referencia para la administración del proyecto. Se centra alrededor de un conjunto de sprints, que son periodos fijos cuando se desarrolla un incremento de sistema".

Schmidt (2016) sostiene que, "es un marco de gestión de proyectos. Scrum especifica ciertos roles, establece un modo de trabajo iterativo que se centra en Sprints y define diferentes artefactos que los desarrolladores utilizan para coordinar su trabajo".

Es un marco de trabajo que está formado por un conjunto de prácticas y reglas que dan respuesta a los siguientes principios de desarrollo ágil: gestión evolutiva del producto, calidad de resultado basado en el conocimiento de las personas, estrategia de desarrollo incremental a través de iteraciones (sprints) (Menzinsky, López, Palacio, 2019).

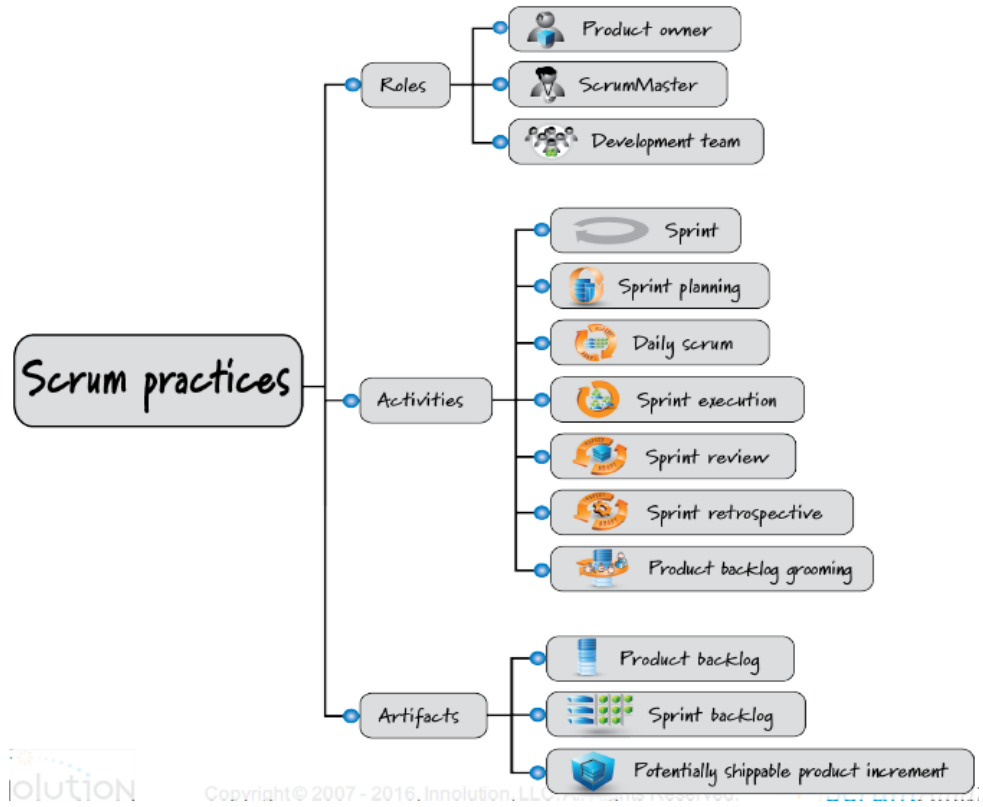


Figura 4. Las Prácticas del Marco de Trabajo Scrum

Fuente: Adaptado de Rubín (2013)

A. LOS VALORES DEL MARCO DE TRABAJO SCRUM

Schwaber y Sutherland (2017) plantean que el marco de trabajo Scrum consta de cinco valores.

a. **CORAJE.** El equipo Scrum tiene el valor de hacer lo correcto y trabajar resolviendo los problemas más difíciles.

b. **FOCUS.** Todo el miembro del equipo Scrum se centra en el trabajo del sprint y los objetivos propuestos.

c. **COMPROMISO.** El equipo Scrum se compromete a alcanzar los objetivos propuestos durante la planificación del sprint.

d. **RESPETO.** Todos los miembros del equipo se respetan recíprocamente por su capacidad e independencia.

e. **APERTURA.** El equipo Scrum y sus partes interesadas acuerdan estar abiertos con todo el trabajo y con los desafíos de su realización.

Figura 5. Los valores de Scrum

Fuente: Adoptado de Schwaber y Sutherland (2017)

A. LOS ROLES EN EL MARCO DE TRABAJO SCRUM

a. Propietario del Producto (Product Owner)

El propietario del producto es responsable de priorizar la pila de productos y de las historias que entran en la pila del Sprint. Debido a esto, él o ella es responsable de aprobar los resultados de cada sprint. El propietario del producto representa directamente o indirectamente los intereses del usuario (Hanssen, Stalhane y Myklebust, 2018).

Pries y Quigley (2011) manifiestan que el propietario del producto representa la voz del cliente. Verifica que el equipo trabaja eficazmente desde el punto de vista empresarial. El propietario del producto se encargará de que el caso de negocio para la pila del producto y los sprints realmente tengan un buen sentido comercial. El propietario del producto se preocupa por los elementos orientados al cliente (historias de usuarios en el desarrollo de software) y luego establecen prioridades y a menudo actualizan la pila de producto.

b. El Scrum Master

Es el "líder servidor" del equipo Scrum que modera y facilita las interacciones del equipo como entrenador de equipo y motivador. El Scrum master

es responsable de asegurar que el equipo tenga un ambiente de trabajo productivo, protegiendo al equipo de influencias externas, eliminando cualquier obstáculo y haciendo cumplir los principios, aspectos y procesos de Scrum (Satpathy, 2016).

El Scrum Master es el facilitador de un equipo que aplica el proceso Scrum al generar valor para el cliente. El Scrum Masters lidera el equipo a través de la planificación, stand-ups diarios, retrospectivas y apoyo a las demostraciones. Al avanzar hacia el enfoque ágil y el cliente, este nuevo rol se centra en la mentalidad ágil, procesos y prácticas ágiles, y la entrega de valor para el cliente. Los Scrum Masters actúan como facilitadores del proceso ágil y líderes del equipo (Moreira, 2017).

c. El equipo de desarrollo (Development Team)

Son un conjunto de profesionales que trabajan para entregar un incremento de producto "Terminado", que pueda ser desplegado en los ambientes de producción al término de cada sprint. Solamente los miembros del equipo de desarrollo intervienen en la construcción del incremento del producto. Los equipos de desarrollo están facultados para autoorganizarse y gestionar su trabajo. La sinergia resultante es la que optimiza la eficiencia y efectividad del equipo de desarrollo (Schwaber y Sutherland, 2017).

Un equipo de desarrollo está formado por un grupo de personas enfocadas en construir el producto o servicio. Se compone de roles multifuncionales como desarrollo, garantía de calidad (QA), base de datos, experiencia de usuario (UX), documentación, etc. Al avanzar hacia el enfoque ágil, los miembros del equipo de desarrollo deben aprender los valores y principios ágiles y aplicar comportamientos y procesos ágiles ya que incorporan las necesidades del cliente en la entrega de valor al cliente. Ellos deben aplicar una mentalidad incremental y procesos que ayuda a evolucionar su entregable hacia el valor del cliente. Aunque técnicamente centrados, deben obtener conocimientos comerciales del propietario del producto para que puedan comprender mejor al cliente y el valor del cliente (Moreira, 2017).

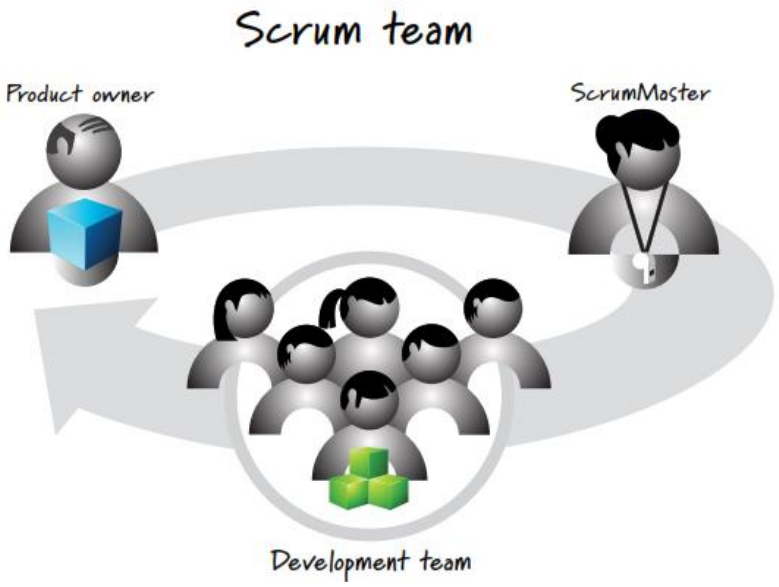


Figura 6. Roles de Scrum

Fuente: Adoptado de Rubín (2013)

B. LAS PRINCIPALES ACTIVIDADES Y ARTEFACTOS DEL MARCO DE TRABAJO SCRUM

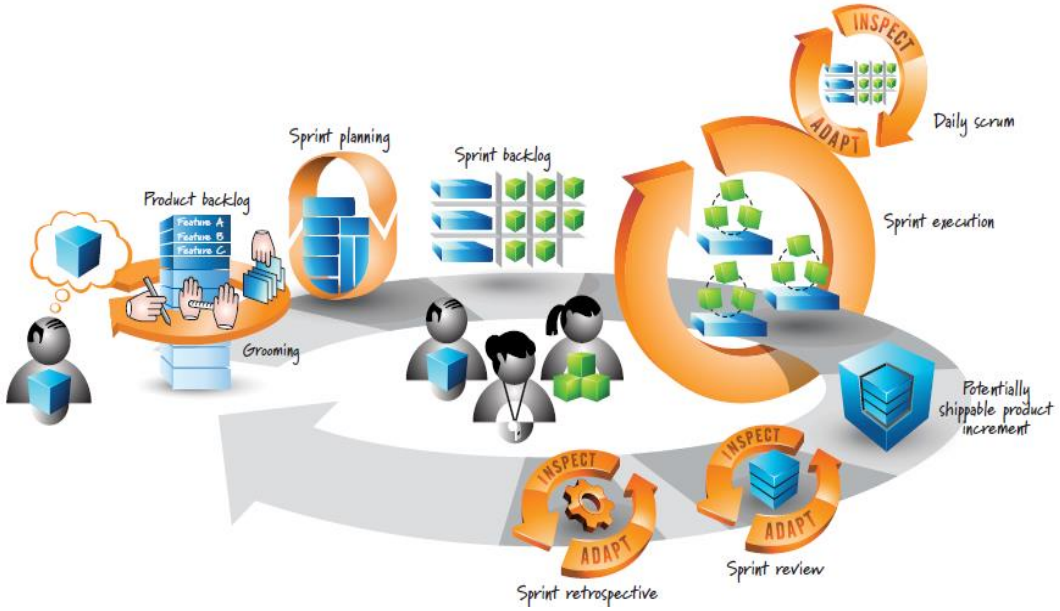


Figura 7. Las principales actividades y artefactos de Scrum

Fuente: adaptado de Rubín (2013)

ACTIVIDADES DEL MARCO DE TRABAJO SCRUM

a. Sprint

Un sprint es un período de desarrollo encuadrado en el tiempo, generalmente de 2 a 4 semanas, donde una parte del código se desarrolla a partir de

un conjunto de historias. Cada sprint genera así un incremento del producto y esta parte se integra con las partes anteriores ya sea al final de el sprint o cuando se ejecutan y aceptan las pruebas relevantes, también llamada "integración continua". De esta manera, el sistema (producto) se construye a través de un proceso iterativo e incremental. El costo total de las historias seleccionadas para un sprint debe ser igual a la cantidad de recursos disponibles para el próximo sprint. La cantidad total de recursos disponibles (horas-persona) es la suma de horas-persona disponibles de los miembros del equipo de desarrollo en el próximo sprint (Hanssen et al., 2018).

Cada sprint es una iteración pequeña y manejable que contiene diseño, desarrollo, pruebas y documentación. La duración de un sprint suele ser de unas dos semanas. El objetivo es producir un entregable para el cliente y poner en producción. Al comienzo de un sprint, el equipo elige los casos de uso más importantes que se puede entregar en la iteración (Stober y Hansmann, 2010).

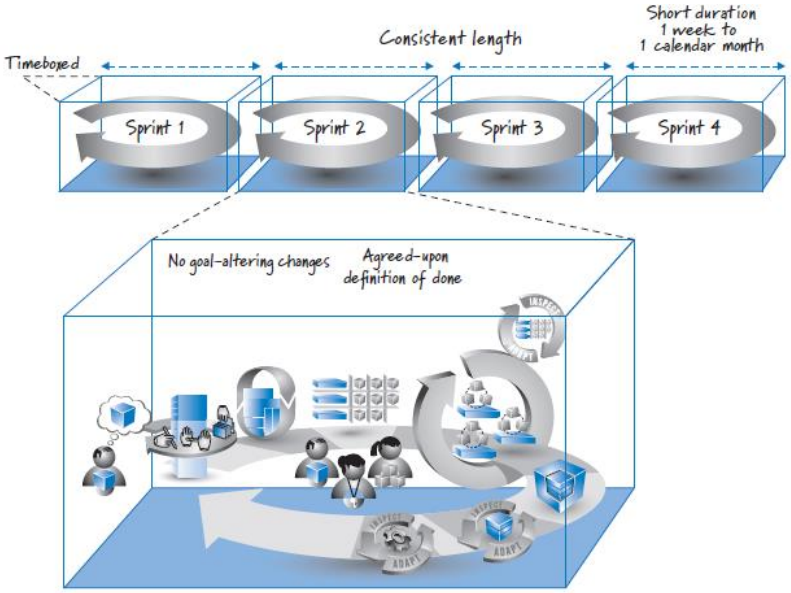


Figura 8. Los sprints representan la estructura de Scrum

Fuente: adaptado de Rubín (2013)

b. Reunión de planificación del Sprint (Sprint Planning Meeting)

La reunión de planificación de sprint es lo primero que sucede en el ciclo y es el inicio oficial de un Sprint. Persigue dos objetivos, que a menudo se representan en dos partes de esta reunión. El primer objetivo es llegar a un acuerdo entre el Propietario del Producto y el equipo de desarrollo lo que se busca en el próximo Sprint. Esto generalmente termina con la estimación que realiza el equipo

de desarrollo, indicando al propietario del producto qué es lo que pueden entregar al final del Sprint actual. El segundo objetivo es que el equipo de desarrollo planifique este siguiente Sprint, a menudo creando tareas de 8 horas o menos. El resultado final de esta reunión es el Sprint Backlog que muestra en una granularidad muy fina lo que el equipo va a hacer y cómo pretenden lograrlo (Maximini, 2015).

Cada sprint comienza con una reunión de planificación de sprints donde los elementos de máxima prioridad de la pila de producto se mueven a la pila del sprint, lo que se suma a la cantidad de recursos disponibles para el sprint. Estos requisitos serán implementados en el siguiente sprint. Cuando una historia de usuario se mueve a la pila del sprint, debe dividirse en tareas. A cada tarea se le asignará una cantidad de recursos. La implementación de las tareas dará cuenta de la historia del usuario (Hanssen et al., 2018).

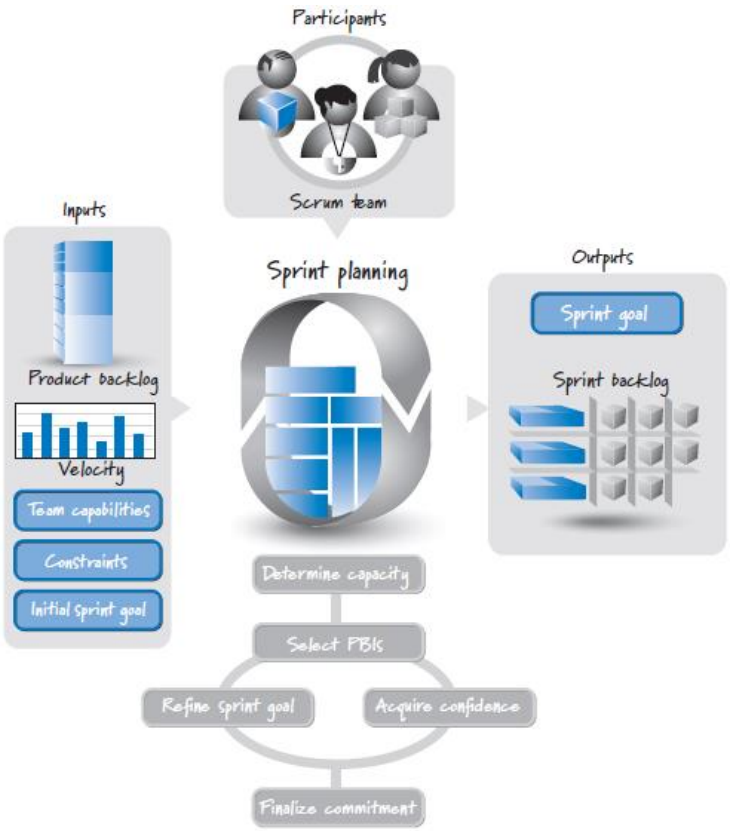


Figura 9. Sprint planning

Fuente: Adaptado de Rubín (2013)

c. Scrum diario (Daily Scrum)

Es una reunión en un intervalo de tiempo de quince minutos para que el equipo de desarrollo pueda realizar la sincronización de sus actividades y crear un

plan para las siguientes veinte cuatro horas. Esto se determina a través de la inspección del trabajo que se realizó en el último Scrum Diario y elaborando una proyección que podría completarse antes del siguiente. En el scrum diario, los equipos suelen responder a las preguntas como ¿Qué hice ayer que ayudó al equipo de desarrollo a alcanzar el objetivo del Sprint? ¿Qué haré hoy para ayudar al equipo de desarrollo a alcanzar el objetivo del Sprint? ¿Veo algún impedimento que me dificulta alcanzar el objetivo del del Sprint? (Schwaber y Sutherland, 2017).

El Scrum diario es una reunión de planificación diaria que consta de 15 minutos. El Scrum diario es una gran oportunidad para observar la motivación y la autoorganización del equipo de desarrollo y es un instrumento que reduce en gran medida el riesgo del proyecto. Si se hiciera con menor frecuencia, tomaría mucho más tiempo identificar desviaciones o que el equipo va en la dirección equivocada (Maximini, 2015).

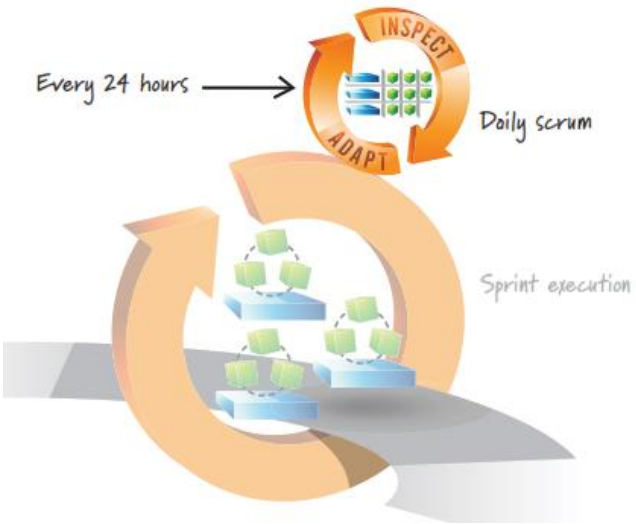


Figura 10. Daily Scrum

Fuente: Adaptado de Rubín (2013)

d. Ejecución del Sprint (Sprint execution)

Es el periodo de tiempo durante el cual el equipo de desarrollo, guiado por el Scrum Master, realiza todo el trabajo a nivel de tarea, necesario para realizar las funciones acordadas durante la planificación del sprint. En este contexto, “terminado” significa que hay un alto grado de confianza en que todo el trabajo necesario para producir características de buena calidad ha sido completado.

Durante la ejecución del sprint, nadie dice al equipo de desarrollo en que orden o como hacer su trabajo a nivel de tarea en el sprint backlog. En cambio, los miembros del equipo definen su propio trabajo a nivel de tarea y se autoorganizan de la mejor manera posible para lograr el objetivo del sprint (Rubin, 2013).

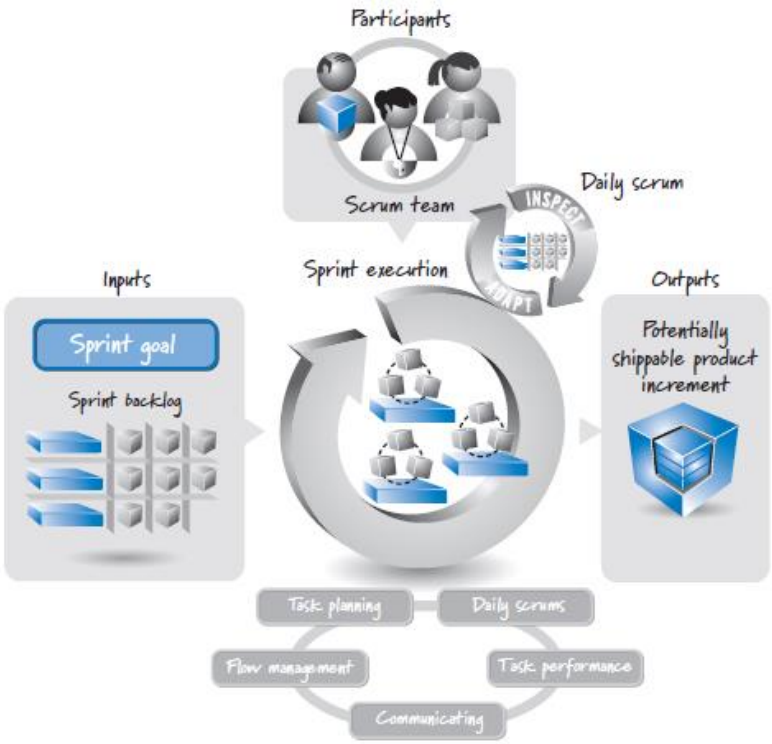


Figura 11. Sprint execution

Fuente: Adaptado de Rubín (2013)

e. La revisión de Sprint (Sprint review)

La revisión del sprint se lleva a cabo antes de la retrospectiva del sprint. Su propósito es que el equipo presente las historias de usuario que ha completado durante el sprint. El equipo, el propietario del producto y Scrum Master están presentes en la revisión, junto con las partes interesadas. La revisión consta de una demostración del software desarrollado. Esta demostración es una oportunidad para que el cliente vea el producto y proporcione comentarios. El objetivo de la revisión es mostrar el software de trabajo real; no debe haber una presentación de diapositivas o masas de preparación para esta revisión. Esta reunión se alinea con

el principio ágil de satisfacer al cliente mediante la entrega temprana y continua de software valioso (Blankenship et al., 2011).

Es una actividad que se realiza al término del sprint, cuyo propósito es la inspección del incremento del producto y adaptar los ítems de la pila del producto. Durante esta actividad, participan el equipo Scrum y los stakeholders inspeccionado las funcionalidades del producto implementado a lo largo del sprint, los asistentes a esta reunión cooperan para establecer las siguientes cosas que se podría hacer para optimizar el valor. El objetivo de la presentación del incremento tiene como objetivo facilitar la realimentación de información al equipo de desarrollo y promover la colaboración (Schwaber y Sutherland, 2017).

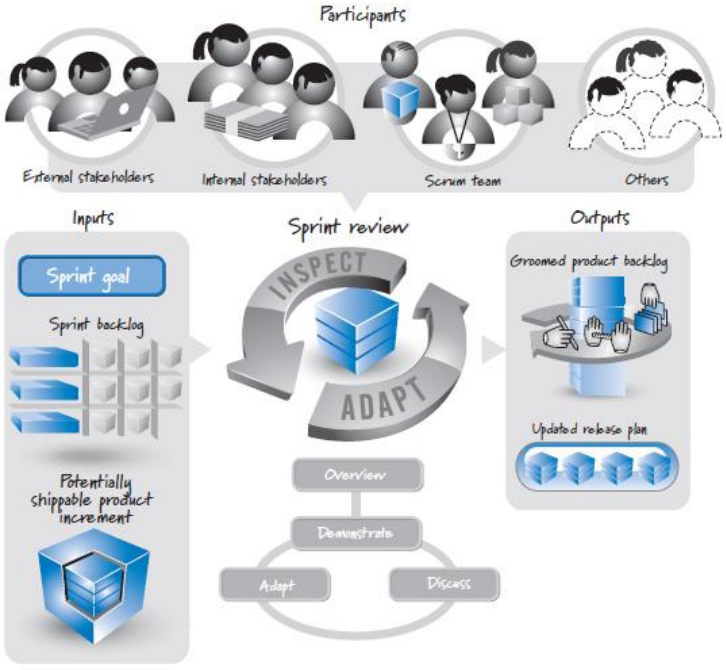


Figura 12. La Revisión del sprint

Fuente: Adaptado de Rubín (2013)

f. La retrospectiva del Sprint (Sprint Retrospective)

Schwaber y Sutherland (2017) manifiestan que es una ocasión para inspeccionar el proceso de construcción del producto software y crear un plan de mejora en el siguiente sprint. La retrospectiva de sprint se realiza después de la revisión del sprint y antes de la siguiente reunión de planificación del Sprint. Se

trata de una reunión, cuya duración es de tres horas para un sprint de un mes, y para sprints más cortos la duración es proporcionalmente menor.

Durante esta reunión, el Scrum Team se reúne para revisar y reflexionar sobre el Sprint anterior en términos de procesos seguidas, herramientas empleadas, mecanismos de colaboración y comunicación, y otros aspectos relevantes para el proyecto. El equipo analiza lo que salió bien durante el Sprint anterior y lo que no salió bien, el objetivo es aprender y hacer mejoras en los próximos sprints (Satpathy, 2016).

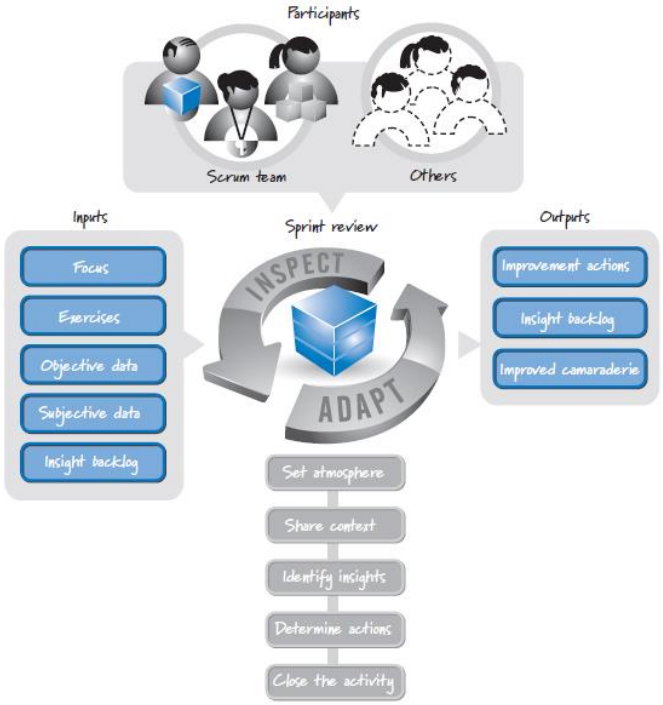


Figura 13. Actividades de la retrospectiva del sprint

Fuente: Adaptado de Rubín (2013)

ARTEFACTOS DEL MARCO DE TRABAJO SCRUM

a. La pila de Producto (Product backlog)

El Product Backlog contiene la suma de todos los requerimientos del producto que deben ser implementados por el equipo de desarrollo. El propietario del producto es el único responsable de gestionar este artefacto. El objetivo principal es que el propietario del producto cree un recordatorio para que el equipo de desarrollo no se olvide de lo que él quiere que hagan. Por lo general, un Product

Backlog no se configura para que se explique por sí mismo. En cambio, una colaboración continua entre propietario del producto y el equipo de desarrollo se aseguran de que este artefacto se entienda comúnmente y se implementan las cosas correctamente. El Product Backlog siempre está ordenado por lo que, en todo momento es transparente lo que se necesita para ser trabajado a continuación. Los elementos que han sido elegidos para el siguiente Sprints, generalmente se entienden bien y son más detallados que los elementos planificados para Sprints posteriores. No hay otras fuentes de trabajo para el equipo de desarrollo (Maximini, 2015).

El Product Backlog es una lista de todo el trabajo restante en un proyecto que el equipo debe completar. En el corazón de esta lista está la historia del usuario, un componente clave de Scrum. Define el incremento de valor para el cliente que el desarrollador está intentando entregar. El Product Backlog es administrada por el propietario del producto, quien es responsable de agregar y eliminar historias de usuarios hacia y desde la lista. El Product Backlog es priorizada constantemente por el propietario del producto y el cliente. Esta priorización constante es la clave de Scrum, porque asegura que las historias de usuario que proporcionen mayor valor al cliente están enumeradas en la parte superior del Product Backlog. Durante un sprint, las historias de usuario se pueden agregar al Product Backlog, sin embargo, no serán presentados al equipo hasta después de que se complete el sprint actual (Blankenship et al., 2011).

b. La pila del sprint (Sprint backlog)

Es la lista de las tareas necesarias para implementar las historias de usuario en un sprint. La elabora el equipo de desarrollo en la reunión de planificación del sprint, indicando el esfuerzo previsto para cada tarea. Para calcular el esfuerzo requerido de cada tarea (puntos de historia o días ideales) es muy frecuente emplear técnicas como la estimación del poker. El sprint backlog descompone las historias de usuario en unidades de tamaño apropiado para monitorizar el avance diario e identificar posibles riesgos y problemas para lograr los objetivos, sin la necesidad de utilizar procesos de gestión complejos (Palacio, 2019).

Representa un subconjunto de historias de usuario que tienen una prioridad lo suficientemente alta como para ser implementados en el sprint. Aquellas historias

que tienen la máxima prioridad y se ajustan a la velocidad del equipo o la cantidad de trabajo que puede completar un equipo dentro de un sprint se convierte en el Sprint backlog. Hay un sprint backlog única para cada sprint. Los elementos del sprint backlog se extrae del Product Backlog durante la planificación del sprint. El Sprint Backlog constituye la columna vertebral del trabajo dentro de un Sprint (Moreira, 2017).

c. El incremento del producto

Schwaber y Sutherland (2017) afirman que "El incremento es la suma de todos los elementos de la lista de productos completados durante un sprint y el valor de los incrementos de todos los sprints anteriores". Al término del sprint, el nuevo incremento debe estar "terminado", esto significa que está en condiciones de ser utilizado y además cumple con la definición de "terminado". El incremento del producto debe estar en condiciones de ser utilizado al margen que el propietario del producto decide liberarlo o no.

"El incremento es la parte de producto producida en un sprint, y tiene como características que está completamente terminada, probada y operativa: en condiciones de ser entregada al cliente final" (Palacio, 2019).

d. Gráfico de avance (Burndown Chart)

Este gráfico es utilizado por el equipo de Scrum para realizar un seguimiento del progreso del Sprint. Las partes interesadas utilizan este gráfico para determinar cuánto trabajo queda (eje y) para ser completado en el Sprint, basado en la cantidad de tiempo restante en el Sprint (eje x). El gráfico muestra la velocidad (rapidez) del equipo cuando está completando el trabajo. Diariamente, el equipo actualiza este cuadro basado en el trabajo que se ha completado. El gráfico también proporciona una indicación de si el equipo completará todo del trabajo planificado en un Sprint (Canty, 2015).

Es un gráfico que muestra la cantidad de trabajo restante en el Sprint en curso. El gráfico de evolución del Sprint debe ser actualizado al final de cada jornada de trabajo, es decir, diariamente a medida que se completa el trabajo. Este gráfico muestra el progreso del proyecto y también permite la detección de estimaciones que pueden haber sido incorrectas. Si el gráfico burn-down muestra

que el que el equipo Scrum no está en camino correcto para terminar las tareas a tiempo, el Scrum Master debe identificar cualquier obstáculo o impedimento para completar con éxito (Satpathy, 2016).

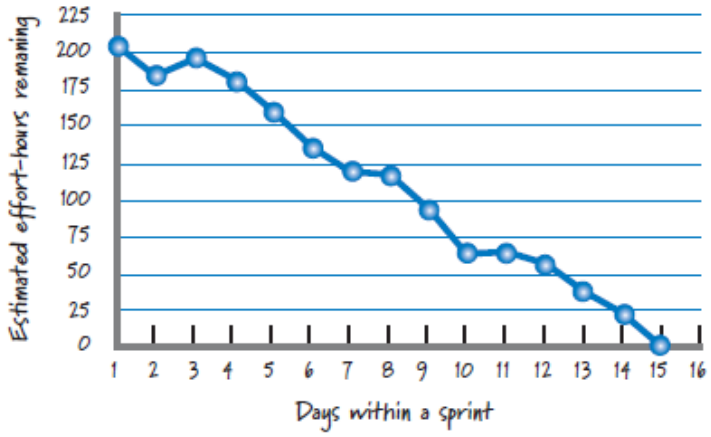


Figura 14. El Gráfico Burndown en la Ejecución del Sprint

Fuente: Adaptado de Rubín (2013)

C. ESTIMACIÓN Y VELOCIDAD EN LAS MÉTODOLOGÍAS ÁGILES

a. LA ESTIMACIÓN DE LOS REQUERIMIENTOS

"Cálculo del esfuerzo que se prevé necesario para desarrollar una funcionalidad. Las estimaciones se pueden calcular en unidades relativas (puntos de función) o en unidades absolutas (tiempo teórico)" (Palacio, 2008).

Pichler (2010) refiere que las estimaciones nos permiten conocer el tamaño aproximado de los ítems del backlog del producto. Es provechoso por dos razones. Facilita la priorización y pronostica el progreso del proyecto para realizar el planificar el lanzamiento. Generalmente se expresan en puntos de historia o días ideales. Las estimaciones a nivel de tarea se crean en la reunión de planificación del sprint.

LAS UNIDADES DE ESTIMACIÓN

Las unidades más comunes son: puntos de historia y días ideales. No existe una alternativa correcta o incorrecta cuando se trata de la elección de una de ellas, la elección depende de cuan familiarizado están sus miembros. El 70% de las

organizaciones suelen utilizar “puntos de historia”, mientras que el 30% utiliza la unidad en “días ideales” (Rubín, 2013).

Estimación por puntos de historia

Cohn (2006) define que “los puntos de historia son una unidad de medida para expresar el total general de una historia de usuario, característica u otra pieza de trabajo”. Cuando se realiza la estimación a través de puntos de historia, se asigna un valor en puntos para cada ítem de la pila del producto. Los valores brutos que se establece no son tan relevantes, porque lo más importante son los valores relativos. Cuando asignamos dos puntos a una historia de usuario, significa que se requiere el doble de esfuerzo que una historia a la que se asignó uno. El tamaño de una historia está representado en puntos de historia. Para determinar el tamaño de una historia de usuario no existe una formula definida.

Estimación por días ideales

“Los días ideales son una unidad familiar: representan la cantidad de días de esfuerzo o días de persona necesarios para completa una historia. El tiempo ideal no es lo mismo que el tiempo transcurrido” (Rubín, 2013).

EL PLANNING POKER

Según Satpathy (2016), es una técnica de estimación que utiliza el consenso para estimar tamaños relativos de las historias de usuario o el esfuerzo necesario para crearlas. Esta técnica intercede por una mayor interacción y una buena comunicación de los integrantes, eso fortalece el razonamiento independiente de los integrantes, eludiendo así el anómalo razonamiento grupal.

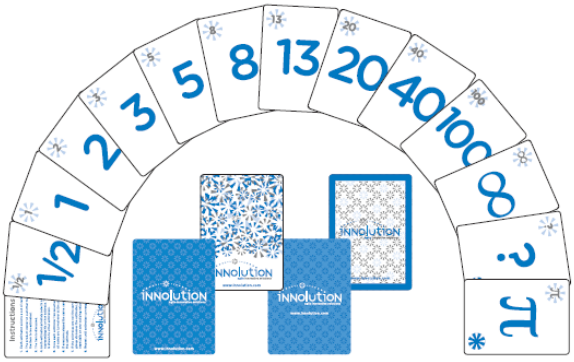


Figura 15. El Juego de Cartas en el Planning Poker

Fuente: Adoptado de Rubín (2013)

Tabla 5

Interpretación de los valores de las cartas en el Planning Poker

Carta	Interpretación de las cartas
1,2,3	Se utiliza para el dimensionamiento de los ítems pequeños.
5,8,13	Es utilizado para dimensionar ítems medianos. Para muchos equipos, un ítem de tamaño trece sería lo más grande que se implementaría en un sprint. Desglosarían cualquier ítem más grande que trece en un conjunto de ítems más pequeños.
20,40	Son utilizados para dimensionar ítems grandes como historias a nivel de tema o de características.
100	Es una característica muy grande o una épica.
?	Significa que un miembro del equipo no ha entendido el ítem y pregunta propietario del producto para que pueda esclarecer. Algunos miembros del equipo también usan el signo de interrogación para indicar que no tienen idea de cómo estimar el ítem en cuestión.
Pi **o una taza de café**	La tarjeta pi se usa cuando un miembro del equipo quiere decir: "Estoy cansado y hambriento y quiero conseguir algo".

Fuente: Adaptado de Rubín (2013)

b. LA VELOCIDAD DEL EQUIPO

"La velocidad es un indicador de cuanto trabajo puede hacer el equipo en un sprint; nos permite rastrear y pronosticar el progreso del proyecto" (Pichler, 2010).

"Es una medida de la tasa de progreso. Se calcula sumando el número de puntos de historia asignados a cada historia de usuario que el equipo completó durante la iteración" (Cohn,2006).

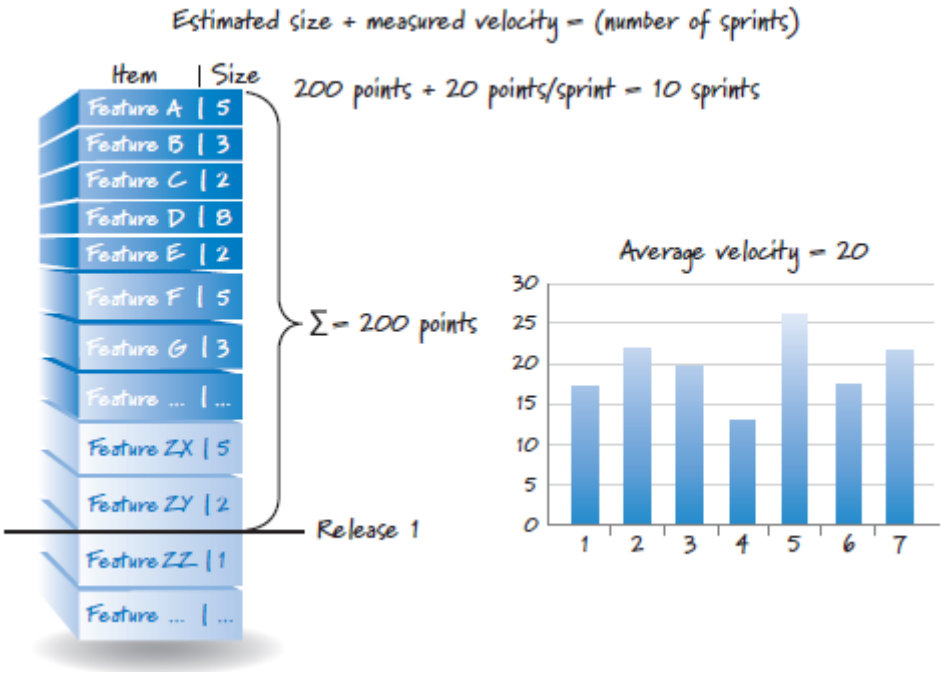


Figura 16. Calcular y usar el rango de velocidades del equipo

Fuente: Adaptado de Rubin (2013)

2.4. HERRAMIENTAS

2.4.1. SOFTWARE JIRA

Sagar (2016) manifiesta que "es una herramienta de gestión de proyectos y seguimiento de problemas empresariales y posee extraordinarias capacidades de personalización". Esta herramienta permite crear épicas, ítems de la pila del producto, tareas, asignarlas a los participantes y generar informes valiosos, además está compuesto por tres componentes y son los siguientes: Jira software, Jira core y Jira servicio.

2.5. POBLACIÓN

Monroy (2008) define como: “la totalidad de los elementos que conforman el universo de estudio. Es el conjunto de valores de una variable por el cual existe algún interés. Cabe agregar que las poblaciones pueden ser finitas o infinitas”.

La población o universo es un conjunto de elementos a los cuales se les estudian algunas características comunes. La población puede ser finita o infinita. Se estima que una población es finita cuando el número de los elementos que la integran es conocido por el investigador, mientras que para la población infinita no se conoce el número de elementos, ya sea porque es muy grande o porque se sabe que existe, pero no se conoce el tamaño (Posada, 2016).

2.6. MUESTRA

Posada (2016) define como un conjunto de elementos seleccionados adecuadamente, que pertenecen a una población determinada, es decir, que es una parte de la población o universo. Al seleccionar una muestra se pretende que el análisis realizado en ella pueda proporcionar conclusiones similares a las que se lograrían si se hubiese estudiado a todos los elementos de la población; por este motivo, la muestra debe ser representativa.

“Es la parte de la población que se selecciona, de la cual realmente se obtiene la información para el desarrollo del estudio y sobre la cual se efectuarán la medición y la observación de las variables objeto de estudio” (Bernal, 2010).

Constituye una selección al azar de una porción de la población, es decir, un subconjunto que seleccionamos de la población. Este grupo reducido de elemento de dicha población, al cual se le evalúan características particulares, con el propósito de inferir tales características a toda la población (Quezada, 2010).

CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1. TIPO Y NIVEL DE INVESTIGACIÓN

A. TIPO DE LA 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). Sobre la base de la consideración anterior, **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). Sobre la base de la consideración anterior, **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). Sobre la base de la consideración anterior, **este trabajo de investigación es de tipo "transversal".**

B. NIVEL DE LA INVESTIGACIÓN

Estos estudios buscan detallar las propiedades, las características y los perfiles de personas, grupos, comunidades, procesos, objetos o cualquier otro fenómeno que se someta a un análisis. En otras palabras, solamente pretenden medir o recolectar información de forma independiente o conjunta referente a los conceptos o las variables. Su objetivo de este tipo de estudio no es indicar como se corresponden estos conceptos o variables (Hernández et al., 2014). De acuerdo a este planteamiento, **la investigación es de nivel "Descriptiva".**

3.2. DISEÑO DE LA INVESTIGACIÓN

Los diseños no experimentales son investigaciones que se efectúan **sin alterar intencionalmente las variables**. O sea, se trata de estudios en los que no realizamos la variación intencional de las variables independientes para reflejar sus efectos sobre las variables dependientes. Lo que se hace en las investigaciones no experimentales es observar el fenómeno en su entorno originario, para posteriormente analizarlo (Hernández et al., 2014). Por esta razón, **el presente estudio es "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). Por esta razón, **el presente estudio es "transversal".**

Este estudio está contextualizado en el **"diseño no experimental"**, puesto que el análisis se realiza a partir de la observación del fenómeno en su contexto natural. Igualmente, se ha considerado el **"diseño transversal"**, de forma que la recolección de datos se efectuará en un solo momento.

3.3. POBLACIÓN Y MUESTRA

A. POBLACIÓN

"La población de estudio estará conformada por todas las metodologías ágiles de desarrollo de software que adoptan los valores y principios del manifiesto agile".

B. MUESTRA

"Se ha tomado una muestra por conveniencia de la metodología ágil de desarrollo de la Programación Extrema (XP) y el marco de trabajo Scrum para la gestión de software ágil".

3.4. VARIABLES E INDICADORES

DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

VARIABLE INDEPENDIENTE

Programación Extrema sobre Scrum. Es un planteamiento para el proceso de desarrollo de software que propone introducir las mejores prácticas de desarrollo de XP sobre el marco de trabajo Scrum, producto de un análisis en el planteamiento de las características similares y orientaciones diferentes que poseen. Estas metodologías se complementan adecuadamente para obtener un incremento del producto de alta calidad a nivel de codificación y la entrega del mismo en la fecha planificada.

INDICADORES DE LA VARIABLE INDEPENDIENTE

Software iterativo e incremental. Es un proceso de desarrollo de software en la que se repite un proceso de trabajo a través de refinamientos sucesivos que adicionan nuevas funcionalidades al código existente.

Software a través de las pruebas unitarias. Son pruebas de software que verifican el resultado de un pequeño fragmento del código sin afectar el funcionamiento de otras funcionalidades del sistema. Una prueba unitaria puede abarcar un método o una clase.

Software a través de la refactorización del código. Es un proceso que se fundamenta en efectuar cambios en la estructura interna de la aplicación sin cambiar su funcionalidad. La refactorización elimina la duplicidad del código, refuerza la cohesión y reduce la sujeción entre los módulos del código.

VARIABLE DEPENDIENTE

Gestión de software ágil. Son una agrupación de actividades que comprende el planeamiento del proyecto, tasación de costos y tiempo, administración de recursos humanos y de riesgos. Es una parte fundamental de la ingeniería de software debido que todos los proyectos están supeditados a las restricciones organizacionales de costos y tiempo.

INDICADORES DE LA VARIABLE DEPENDIENTE

Planificación del sprint. Es una reunión que se realiza al principio de cada sprint con la participación de todos los miembros del equipo Scrum y tiene como propósito establecer objetivos que se entregarán en el presente sprint en base a las estimaciones realizadas. Las estimaciones se realizan a nivel de la pila del producto y sprint backlog en base al grado de dificultad, riesgo y tiempo de culminación.

Ejecución del Sprint. Es el periodo de tiempo durante el cual el equipo Scrum realiza el incremento del producto a partir de los requerimientos acordados en la planificación del sprint. Durante este proceso se emplean con mayor amplitud las mejores prácticas de desarrollo.

Retrospectiva del Sprint. Es una reunión que consiste en la inspección y adaptación del proceso que se ha utilizado para construir el producto software. La retrospectiva permite realizar un análisis exhaustivo de los métodos empleados, capitalizar aprendizajes y definir acciones de mejora para el próximo sprint.

DEFINICIÓN OPERACIONAL DE LAS VARIABLES DE ESTUDIO

VARIABLE INDEPENDIENTE DEL ESTUDIO

X: “Programación Extrema sobre Scrum”

INDICADORES DE LA VARIABLE INDEPENDIENTE

X_1: Software iterativo e incremental

X_2: Software a través de las pruebas unitarias

X_3: Software a través de la refactorización del código.

VARIABLE DEPENDIENTE

Y: “Gestión de software ágil”.

INDICADORES DE LA VARIABLE DEPENDIENTE

Y_1: Planificación del Sprint.

Y_2: Ejecución del Sprint.

Y_3: Retrospectiva del Sprint.

3.5. TÉCNICAS E INSTRUMENTOS PARA EL TRATAMIENTO DE DATOS E INFORMACIÓN

3.5.1. TÉCNICAS PARA RECOLECTAR INFORMACIÓN

En el presente estudio utilizaremos la técnica "Análisis Documental" que permitirá realizar la recolección de la información sobre la "metodología ágil de la programación extrema" y el "marco de trabajo Scrum".

3.5.2. INSTRUMENTOS PARA RECOLECTAR INFORMACIÓN

Ficha de análisis documental. Instrumento que nos permite levantar el contenido de la información sobre la "metodología ágil de la programación extrema" y el "marco de trabajo Scrum", para posteriormente someter a un proceso de análisis e interpretación riguroso. Se muestra en el Anexo A.

3.5.3. HERRAMIENTAS PARA EL TRATAMIENTO DE DATOS E INFORMACIÓN

Las herramientas que se emplearán para la implementación del "Modelo de desarrollo de software de la Programación Extrema sobre Scrum para permitir la gestión de software ágil" son: "Metodología ágil de la Programación Extrema" que utiliza las mejores prácticas técnicas de desarrollo y el "marco de trabajo Scrum" que enfatiza en las prácticas de gestión y organización, asimismo tenemos a la herramienta Jira para la administración de los ítems de la pila del producto y las tareas de un proyecto software.

Tabla 6

Herramientas para la implementación del modelo XP sobre Scrum

NOMBRE	CREADOR	USO
Metodología de software ágil Programación Extrema (XP)	Kent Beck	Es una metodología de desarrollo de software ágil que se centra más en las buenas

		prácticas de desarrollo y pruebas.
Marco de trabajo Scrum	Ken Schwaber y Jeff Sutherland	Es un marco de trabajo que nos provee prácticas para organizarnos como equipo y gestionar los objetivos de un proyecto software.
Jira	Atlassian Labs	Es una herramienta de gestión de requisitos que se utiliza en un proyecto de desarrollo de software ágil.

Fuente: Elaboración propia

3.6. PROPUESTA DEL MODELO DE DESARROLLO DE SOFTWARE DE LA PROGRAMACION EXTREMA SOBRE SCRUM

3.6.1. ANÁLISIS COMPARATIVO

Para obtener el "Modelo de desarrollo de software de la Programación Extrema sobre Scrum para el desarrollo de software", urge un análisis meticuloso sobre el contenido de la "metodología ágil de la Programación Extrema" y "marco de trabajo Scrum". Estas metodologías tienen un aspecto en común, ya que adoptan los 4 valores y 12 principios de la agilidad, sin embargo, el primero está más orientado en las prácticas técnicas de desarrollo, el segundo, tiene un enfoque en las prácticas de organización y gestión durante el desarrollo de un proyecto software. "El Modelo de desarrollo de software de la Programación Extrema sobre Scrum", busca obtener un producto software con altos estándares de alta calidad y escalable. Para lograr un producto software con esta característica, planteamos adoptar estos dos componentes esenciales: Las mejores prácticas de desarrollo proporcionado por XP y el enfoque de organización y gestión proporcionado por el marco de trabajo Scrum. A continuación, realizaremos un análisis comparativo de las diferencias y similitudes de la metodología ágil de la Programación Extrema y el marco de trabajo Scrum a través de las matrices de comparación.

Tabla 7

Cuadro comparativo de Scrum y XP

Parámetros	Metodologías Ágiles de Desarrollo	
	Scrum	**Programación Extrema**
Orientación	Prácticas de organización y gestión	Técnicas de desarrollo y pruebas de software
Coordinador del proyecto	Scrum master	Entrenador
Forma de trabajo del equipo de desarrollo	Individual	En pareja
Gestión de requerimientos	Requerimientos gestionados en forma de artefactos a través de Product backlog y Sprint backlog	Gestionados en forma de tarjetas de historia
Propiedad del producto	El Propietario del producto es el responsable del producto	Responsabilidad grupal de producto
Colaboración en equipo	Equipos multifuncionales y auto organizados	Equipos auto organizados
Entrega del producto	Entrega según el timebox del sprint	Entrega continua
Estándares de codificación	Sin estándares de codificación	Se utilizan estándares de codificación
Enfoque de pruebas	Ningún enfoque formal utilizado para las pruebas	Desarrollo dirigido por pruebas, incluida pruebas de aceptación
Cambios de desarrollo	Cambios no permitidos en sprints	Es posible realizar cambios durante la iteración

Fuente: Elaboración propia

Tabla 8
Valores de Scrum y XP

Parámetro	Scrum	XP	Comparación
Valores	Coraje	Coraje	Valores similares
	Respeto	Respeto	
	Compromiso		Los siguientes valores son diferentes
	Enfoque		
	Franqueza		
		Simplicidad	
		Realimentación	
		Comunicación	

Fuente: Elaboración propia

Tabla 9

Roles de Scrum y XP

Parámetro	Scrum	XP	Comparación
Roles	Propietario del producto	Cliente	Roles similares
	Scrum master	Entrenador	
	Equipo de desarrollo	Programador Encargado de pruebas	Estos roles de XP están incluidos en rol de equipo de desarrollo de Scrum
		Encargado de seguimiento Consultor	Estos roles son propios de la programación extrema.

Fuente: Elaboración propia

Tabla 10
Artefactos de Scrum y XP

Parámetro	Scrum	XP	Comparación
Artefactos	Product Backlog (Pila del producto)	Historias de usuario	Artefactos similares
	Sprint Backlog (Pila del Sprint)	Tareas de ingeniería	
	Incremento del producto		Los siguientes artefactos serán integrados en el nuevo modelo de desarrollo.
	Burndown chart (Seguimiento del avance)		
		Casos de pruebas de aceptación	
		Tarjetas CRC	

Fuente: Elaboración propia

Tabla 11
Actividades de Scrum y XP

Parámetro	Scrum	XP	Comparación
Actividades	Sprint	Planificación de iteraciones	Actividades similares
	Planificación del Sprint (Sprint Planning)	Juego de planificación (Planning Game)	
	Reunión diario (Daily Scrum)	Stand-up meeting (reunion de pie)	
	Ejecución del Sprint (Sprint execution)		Estas actividades serán plasmadas en el modelo propuesto.
	Revisión del Sprint (Sprint review)		
	Retrospectiva del Sprint (Sprint retrospective)		

Fuente: Elaboración propia

Tabla 12
Prácticas y principios de Scrum y XP

Parámetro	Scrum	XP	Comparación
Prácticas y principios		Entregas pequeñas	Estas metodologías agiles, poseen prácticas y principios que pueden complementarse para desarrollar un producto software con las mejores prácticas de desarrollo y pruebas, propuesto por la programación extrema y las prácticas de gestión y organización propuesto por Scrum.
		Pruebas	
		Diseño simple	
		Programación por parejas	
		Refactorización	
		Desarrollo dirigido por pruebas	
		Integración continua	
		Propiedad colectiva del código	
		Estándares de codificación	
		Metáfora del sistema	
		Ritmo sostenible	
	Control empírico de proceso		
	Colaboración		
	Bloque de tiempo asignada		
	Auto organización		
	Priorización basada en el valor		

Fuente: Elaboración propia

3.6.2. ESTUDIO DE INTEGRACIÓN DE LA PROGRAMACIÓN EXTREMA SOBRE SCRUM

La incorporación de los valores, roles, artefactos, actividades y las prácticas de desarrollo de la metodología ágil de la Programación Extrema al marco de trabajo Scrum, nos permitirá obtener un producto software confiable, robusto y escalable y funcional. A partir del estudio de comparación, se puede inferir que el marco de trabajo Scrum se complementan perfectamente con la metodología ágil de la Programación Extrema. A través de la matriz de integración, obtendremos las características más relevantes de XP para incorporar al marco de trabajo Scrum denominado "Modelo de desarrollo de desarrollo de software de la Programación Extrema sobre Scrum".

Tabla 13
Integración de los valores en el modelo propuesto

Indicador	Modelo propuesto	Observación
Valores	Coraje	Valores pertenecen a Scrum. Serán incorporados al modelo propuesto.
	Respeto	
	Compromiso	
	Atención	
	Franqueza	
	Simplicidad	Valores que pertenecen a XP. Serán incorporados al modelo propuesto.
	Realimentación	
	Comunicación	

Fuente: Elaboración propia

Tabla 14
Integración de los roles en el modelo propuesto

Indicador	Modelo propuesto	Observación
Roles	Dueño del producto	No se agregaron roles de XP. El equipo de desarrollo adopta la auto organización y la colaboración para generar el incremento.
	Scrum master	
	Equipo de desarrollo	

Fuente: Elaboración propia

Tabla 15
Integración de los artefactos en el modelo propuesto

Indicador	Modelo propuesto	Observación
Artefactos	Product Backlog (Pila del producto Historias de usuario	Este artefacto pertenece a Scrum. Será incorporado al modelo propuesto.
	Sprint Backlog (Pila del Sprint)	El sprint backlog está constituido por un conjunto de tareas a nivel de desarrollo.
	Incremento del producto	Estos artefactos pertenecen a Scrum. Será incorporado al modelo propuesto.
	Burndown chart (Seguimiento del avance)	
	Pruebas de aceptación	Este artefacto pertenece a XP. Será incorporado al modelo propuesto.

Fuente: Elaboración propia

Tabla 16
Integración de actividades en el modelo propuesto

Indicador	Modelo de desarrollo propuesto	Observación
Actividades	Sprint	Estas actividades pertenecen a Scrum. Serán incorporados al modelo propuesto.
	Sprint Planning (Planificación del Sprint)	
	Daily Scrum (Reunión diaria)	
	Sprint execution (Ejecución del Sprint)	Esta actividad será incorporada al modelo propuesto
	Sprint review (revisión del Sprint)	Estas actividades pertenecen a Scrum. Serán incorporados al modelo propuesto.
	Sprint retrospective (Retrospectiva de Sprint)	

Fuente: Elaboración propia

Tabla 17
Integración de las prácticas y principios en el modelo propuesto

Indicador	Modelo de desarrollo propuesto	Observación
Prácticas y principios	Entregas pequeñas	Estas prácticas y principios pertenecen a XP y serán incorporados al modelo propuesto.
	Pruebas	
	planificación	
	Diseño simple	
	Programación por parejas	
	Refactorización	
	Desarrollo dirigido por pruebas	
	Integración continua	
	Propiedad colectiva del código	
	Estándares de codificación	
	Ritmo sostenible	
	Control empírico de proceso	Estas prácticas y principios pertenecen al Scrum. Serán incorporados al nuevo propuesto.
	Colaboración	
	Bloque de tiempo asignada	
	Auto organización	
	Priorización basada en el valor	

Fuente: Elaboración propia

3.6.3. VISIÓN GENERAL DEL MODELO DE DESARROLLO DE LA PROGRAMACIÓN EXTREMA SOBRE SCRUM

El nuevo "Modelo de desarrollo de software de la programación extrema sobre Scrum" muestra una visión general de las principales actividades, artefactos y prácticas en la Figura 17. Son cuatro las principales actividades y son los siguientes: Planificación, ejecución, revisión y la retrospectiva del sprint. Sin embargo, cada una de estas actividades mencionadas a su vez poseen otras subactividades, artefactos, prácticas, etc.

La planificación está compuesta por la estimación de los ítems del backlog del producto y la planificación del sprint backlog; en la primera reunión participan

todos los miembros del equipo Scrum, mientras que en la segunda reunión solo participan el Scrum Master y el equipo de desarrollo. Para la estimación de los ítems del backlog del producto y las tareas se utiliza la técnica del planning poker.

La ejecución del sprint está compuesta sobre todo por las buenas prácticas de desarrollo, además se utiliza el gráfico burndown y el scrumboard o tablero de tareas. Durante esta actividad se produce el incremento del producto.

Revisión del sprint es una reunión para la inspección y adaptación del incremento del producto.

Retrospectiva del sprint es una reunión para inspeccionar y adapar el proceso que se ha empleado para construir el producto software.

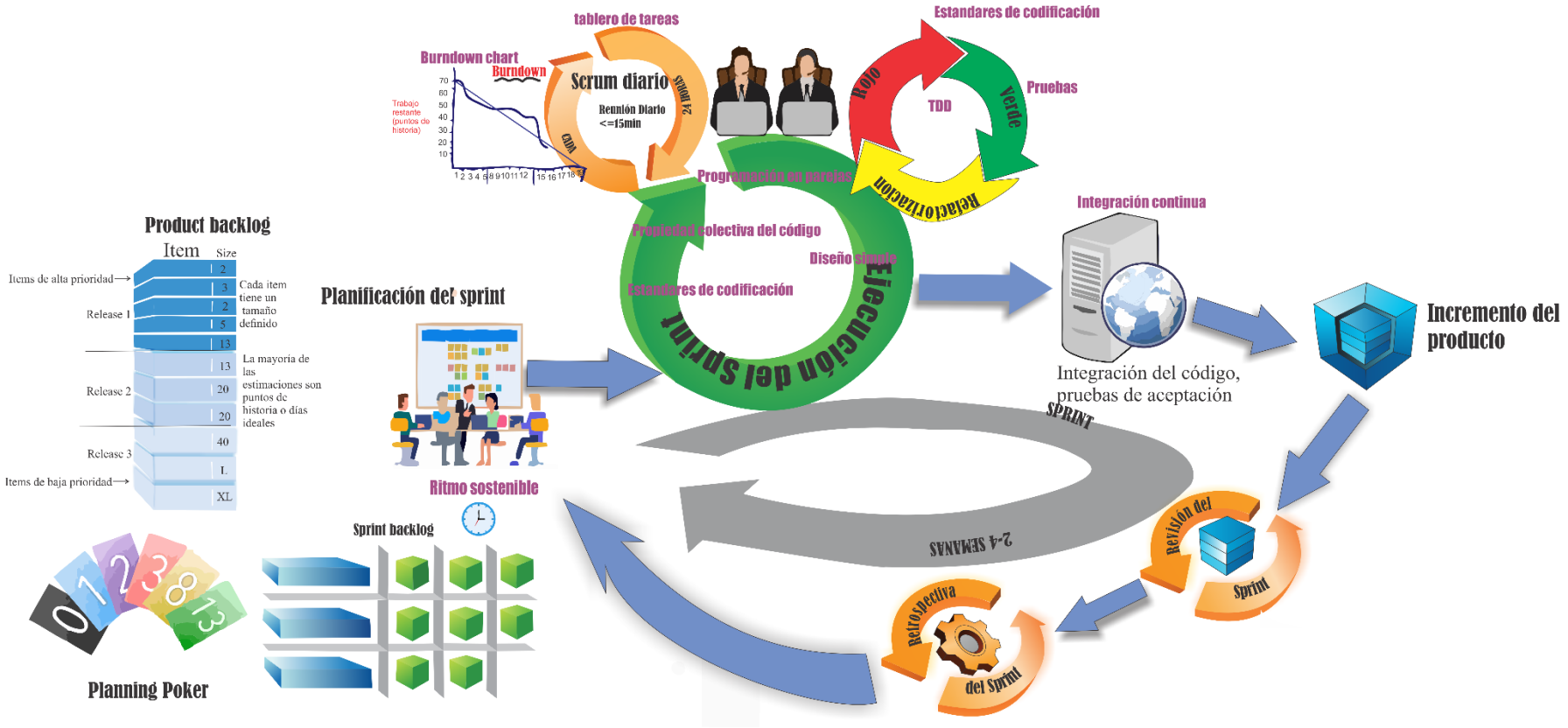


Figura 17. Visión general del “Modelo de desarrollo de software de la programación extrema sobre Scrum”

Fuente: Elaboración propia

3.6.4. PROPUESTA DEL NUEVO MODELO DE DESARROLLO DE LA PROGRAMACION EXTREMA SOBRE SCRUM

3.6.4.1. Fase de preparación (Sprint cero)

Como afirma Subra y Vannieuwenhuyze (2018), es un periodo de preparación inicial, bien determinado en el tiempo, entendiendo que esto permitirá facilitar el desarrollo de los sprints, por ejemplo, para establecer todos los elementos de la plataforma técnica y las herramientas de desarrollo, pruebas e integración continua. Durante este periodo se pueden realizar las siguientes actividades:

- Definir la arquitectura.
- Definir las pruebas a realizar.
- Preparar y configurar los repositorios de control de versiones.
- Configurar los scripts para el servidor de integración continua.
- Escribir las historias de usuario para realizar la estimación en el proceso de planificación del sprint.
- Definir herramientas de gestión que utilizará el equipo.
- Definir estándares de codificación.

Historia de usuario

"Una historia de usuario describe la funcionalidad que será valiosa para el usuario u comprador de un sistema software" (Cohn,2009).

"Las historias de usuario representan los medios para obtener una comprensión común de la funcionalidad entre las partes interesadas" (Canty, 2015).

A juicio de Wake (2003), "los criterios que definen los atributos de una historia de usuario efectiva identificada con el acrónico INVEST" y son los siguientes:

- **Independent**. No debe existir una dependencia entre historias de usuario
- **Negotiable.** Una buena historia deja espacio para que las partes involucradas negocien los detalles de su implementación
- **Valuable.** Una buena historia de usuario debe ser valiosa para el cliente. Debe describir una característica o un servicio que el cliente necesita. Debido a que las historias de usuario se programarán e implementarán en

una iteración, deben agregar valor para el producto cuando se completa la iteración.

- **Estimable.** Las historias de usuarios deben poder estimarse. La estimación es fundamental para el propietario del producto pueda asignar una prioridad relativa a la historia. Esta característica también permite que el equipo se concentre en el tamaño de la historia.
- **Small.** El tamaño de las historias de usuario es importante, porque si son demasiado grandes o demasiado pequeñas, no se pueden utilizar adecuadamente en la planificación.
- **Testable.** Las historias de usuario deben estar escritas de manera que sean comprobables. Pasar con éxito sus pruebas demuestra que una historia se ha desarrollado con éxito.

La plantilla de la historia de usuario se encuentra en el Anexo B.

3.6.4.2. Reunión de planificación del sprint

Es una de las primeras actividades que se realizan al principio de cada sprint. En esta reunión participan: el Propietario del Producto (Product Owner), el Scrum master y el equipo de desarrollo (Development team).

El objetivo de esta reunión, es que el propietario del producto pueda presentar al equipo de desarrollo, los ítems del backlog del producto prioritarias y de mayor valor que se implementarán en el presente sprint, estos ítems deben estar incluidas en el backlog del producto; y que el equipo comprenda el alcance de las mismas mediante interrogantes al propietario del producto, para luego realizar la estimación del esfuerzo requerido para completar cada uno de los ítems en el presente sprint. La duración del sprint, también se determina durante esta reunión de planificación del sprint.

La estimación se realizará en dos momentos. La primera parte de la reunión consiste en la estimación de los ítems del backlog del producto y la segunda parte consiste en la estimación del sprint backlog. En la primera parte planteamos la siguiente interrogante ¿qué construir?, donde el equipo de desarrollo en base a su capacidad; es decir, la velocidad del equipo y la estimación del esfuerzo requerido de los ítems del backlog del producto (en puntos de historia) se compromete

entregar ciertas funcionalidades del producto al finalizar el sprint. Para la segunda parte de la reunión se plantea la interrogante ¿Cómo construir? El equipo fragmenta los ítems de la pila del producto en un conjunto de tareas denominado sprint backlog para realizar la estimación correspondiente (en horas) del esfuerzo requerido para completar cada una de estas tareas.

a. Primera parte: Estimación del Produc backlog ¿Qué construir?

En esta primera parte de la reunión es sustancial realizar la estimación de los ítems de la pila del producto con la participación de todo el equipo Scrum, para tal fin emplearemos una técnica muy común que se acostumbra utilizarse en las diferentes metodologías ágiles de desarrollo, nos referimos al "Planning Poker".

Planning Poker

Para utilizar esta técnica de estimación, es importante la participación de todo el equipo Scrum. Los miembros participantes desempeñan diversas funciones. El propietario del producto es el que exhibe, detalla y dilucida los ítems de la pila del producto. El Scrum Master explica las reglas antes de iniciar con la estimación; durante el proceso lidera aplicando de forma eficaz el Planning Poker, además identifica a los participantes que muestran un comportamiento sigiloso o parezcan estar disidentes y le apoya para que puedan participar. Este último debe participar de forma muy activa, puesto que, son los encargados de realizar la estimación de los ítems de la pila del producto.

Reglas del Planning Poker

Bahit (2012) propone las siguientes reglas del Planning Poker:

Antes de iniciar

- Se debe tomar una decisión con respecto al tiempo de explicación atribuido a cada uno de los participantes para que puedan exponer su estimación.
- Además, se debe aclarar con respecto al receso, cuando uno de los participantes muestre la carta "taza de café".

Durante el juego

Una vez que el propietario del producto presenta un ítem de la pila del producto, los miembros del equipo de desarrollo plantean las interrogantes correspondientes para poder comprender la magnitud del mismo, es el momento para establecer el esfuerzo y la dificultad que demanda la construcción de una "historia de usuario". En este momento empieza el juego.

1. Cada uno de los miembros del equipo de desarrollo deben tener en su poder un juego de cartas.
2. Tras el debate del alcance, cada miembro del equipo de desarrollo piensa en el esfuerzo necesario que se necesitará para la construcción de la funcionalidad demandada, y procede a colocar la carta correspondiente sin mostrar el número el numero de la carta sobre la mesa.
3. Cuando todos los participantes hayan colocado una carta sobre la mesa, se debe mostrar la estimación asignada.
4. En el caso de que los participantes arrojasen la misma carta, la estimación culmina y se debe asignar a la historia de usuario correspondiente. Una vez culminado, todos los participantes levantan sus cartas de la mesa y continúa la estimación del siguiente ítem de la pila del producto (Se repite desde el paso 2).
5. Si se presentan discrepancias en la estimación de los ítems, el participante que mayor esfuerzo haya estimado debe fundamentar sus motivaciones, posiblemente haya encontrado algunas dificultades que los otros participantes no habían tomado en cuenta. En seguida, explicará el miembro que menor esfuerzo haya estimado, indudablemente, ha encontrado una forma más fácil de solucionar a diferencia de los otros participantes.
6. Culminada la explicación de los miembros participantes, se vuelve a estimar el ítem, repitiendo la secuencia desde el paso dos.

Después de haber realizado la estimación de los ítems de la pila del producto utilizando la técnica del Planning Poker, procedemos a seleccionar los

ítems de mayor prioridad, ubicando en la parte superior de la pila hasta extinguir la capacidad del equipo de desarrollo. La capacidad del equipo está estrechamente relacionada con la velocidad del equipo que consiste en la construcción de cierta cantidad de los ítems en un sprint y otros factores externos que puedan presentarse. Para obtener la velocidad del equipo verificaremos la información histórica de los sprints anteriores, en caso de la inexistencia de esta información histórica procedemos a utilizar lo que comúnmente se utiliza, la "velocidad relativa".

Para lograr los objetivos propuestos, el equipo de desarrollo descompone cada ítem de la pila del producto en un conjunto de tareas, denominado sprint backlog. Cada una de estas tareas son estimada en horas del esfuerzo requerido para culminar cada una de las tareas. Descomponer los ítems en un conjunto de tareas es una forma planificación justo a tiempo, asimismo nos permite trabajar a un ritmo que puede sostenerse en el tiempo.

Backlog del producto

Es un listado dinámico, público y perceptible de los requerimientos funcionales que son actualizados constantemente por el propietario del producto. Los requerimientos en los que laboraremos en el siguiente sprint deben caracterizarse por ser pequeños y detallados, no obstante, los requerimientos que serán implementados posteriormente pueden ser refinados gradualmente en un conjunto de ítems más pequeños y precisos conjuntamente con la parte interesada, propietario del producto y los miembros del equipo de desarrollo. Un ítem de la pila del producto se caracteriza por ser lo suficientemente pequeño y detallado para ser diseñado, construido y probado en un sprint.

La plantilla base de registro de los ítems del backlog del producto se encuentra en el Anexo C.

Sprint

Una de las características importantes del sprint, es que están determinados por fechas definidas de inicio y finalización fijas, llamado "timebox". Se sugiere trabajar en sprints de corta duración, específicamente, entre dos semanas o un mes calendario como máximo. "Los sprint de corta duración brindan muchos beneficios

como la facilidad de planificación, retroalimentación rápida, mejor retorno de inversión, error acotado, entusiasmo rejuvenecido, conjunto bien definido de hitos y la duración consistente" (Rubín, 2013). Con referencia a lo anterior sugerimos que se debe adoptar un sprint de dos semanas (15 días). Asimismo, como regla general, no se admite cambios que afecten el objetivo durante un sprint que ya empezó, y, por último, cabe resaltar que se debe entregar el incremento del producto de acuerdo a la definición del término "hecho" o "terminado".

Terminado ("Done")

Son aquellos requisitos que un ítem del backlog del producto y todo el incremento deben cumplir para aseverar que el trabajo de desarrollo asociado a finalizado y, en consecuencia, pueda ser entregado. Por lo tanto, es primordial que todos los miembros del equipo tengan un entendimiento compartido del significado de "terminado". Como afirma Rubín (2013), "El incremento del producto se compone de un conjunto de ítems del backlog del producto, por lo que cada ítem del backlog del producto debe completarse de conformidad con el trabajo especificado por la definición lista de verificación de terminado".

En base a la conceptualización anterior, proponemos un conjunto de requisitos que debe cumplir cada uno de los ítems del backlog del producto y todo el incremento, para confirmar que ya fueron culminados. Estos requisitos son los siguientes:

i. Diseño revisado

ii. Código completado

- Código con estándares de codificación
- El código debe tener comentarios
- Código verificado (sin errores)
- Código inspeccionado
- Código refactorizado

iii. Documentación actualizada para el usuario final

iv. Pruebas

- Pruebas unitarias
- Pruebas de integración

- Pruebas de aceptación

v. Desplegar en producción

b. Segunda parte: Estimación del Sprint Backlog ¿Cómo construir?

En esta actividad participan el Scrum Master y el equipo de desarrollo, y consiste en dividir los ítems de la pila del producto en un conjunto de tareas, llamado Sprint Backlog, para luego realizar la estimación, en horas de esfuerzo, para completar cada tarea. El equipo de desarrollo procede a comparar la estimación realizada de los ítems de la pila del producto y sus respectivas tareas para que pueda alcanzar los objetivos del siguiente sprint. Si el equipo ha seleccionado ítems del backlog del producto que no puede desarrollar de manera realista y objetiva, puede refinar los objetivos propuestos a nivel de los ítems de la pila del producto, para que se ajuste a la capacidad disponible del equipo y posibles limitaciones.

Sprint Backlog

El sprint backlog es una lista de tareas a nivel de desarrollo de software estimadas en no más de ocho horas por el equipo de desarrollo. La estimación de estas tareas se realiza en la segunda parte de la reunión de planificación, esta lista únicamente podrá ser gestionado por el equipo de desarrollo a lo largo del sprint. Esta lista se genera para cada uno de los ítems de la pila del producto planteados en la primera parte de la reunión de planificación y representa a los requerimientos que se implementarán a lo largo del próximo Sprint.

La plantilla de registro del sprint backlog se muestra en el Anexo D.

Ritmo sostenible

Esta práctica, implica trabajar una semana laboral estándar, o sea, 40 horas semanales y no emplear horas extras para evitar el agotamiento y el estrés que puedan sufrir los miembros del equipo de desarrollo que participan en el proyecto.

El ritmo sostenible se logra a través de una planificación adecuada, puesto que las horas extras son el efecto directo de una inadecuada planificación y el limitado compromiso que asume el equipo durante esta etapa. Desde momento en que el equipo coopera de manera responsable en la estimación de los ítems de la

pila del producto y posteriormente en las tareas de la pila del sprint, se compromete a lograr los objetivos en base a su capacidad, y al utilizar las herramientas como el gráfico de burndown y el tablero para verificar el progreso del trabajo ocasiona una métrica amplia y perceptible que evita sobresaltos en la fecha de finalización del sprint, en consecuencia, prescinden de las horas extras.

3.6.4.3. Ejecución del Sprint

Es el trabajo que realizan los miembros del equipo Scrum para cumplir los objetivos establecidos durante la reunión de planificación del sprint, es decir, construir y entregar el producto software completamente terminada y operativa al final del sprint. Para alcanzar este objetivo, el equipo de desarrollo debe practicar el principio de autoorganización. Además, de las buenas prácticas de desarrollo y pruebas de la metodología ágil de la Programación Extrema. Estas prácticas de XP son los siguientes: Programación en pareja, propiedad colectiva del código, diseño simple, metáfora del sistema, pruebas unitarias, pruebas de aceptación e integración continua. Esta actividad, también se complementa del Scrum diario, propuesto por el marco de trabajo Scrum.

Programación en pareja

Permite que dos desarrolladores trabajen juntos en una determinada tarea compartiendo un solo computador. Se sugiere que estos desarrolladores deben cambiar de pareja como parte de la implementación de diferentes tareas del proyecto y el aprovechamiento de conocimientos mutuamente entre un desarrollador experto y otro menos experto. Mediante esta práctica podemos reforzar aún más la propiedad colectiva, difundiendo los módulos del sistema en torno a todo el equipo.

Scrum diario

“El Scrum diario consiste en la inspección, sincronización y planificación adaptativa diaria, esta actividad ayuda a un equipo auto organizado a realizar mejor su trabajo”.

Durante esta actividad se reúnen el Scrum Master y los miembros del equipo de desarrollo por un intervalo de quince minutos. En esta reunión, cada

miembro del equipo de desarrollo explica brevemente la respuesta a las siguientes preguntas:

a. ¿Qué hice ayer para lograr el objetivo del sprint?
b. ¿Qué haré hoy para lograr el objetivo del sprint?
c. ¿Qué es lo que me impide lograr el objetivo del sprint?

Para responder estas interrogantes, es necesario contar con los siguientes artefactos: Scrum taskboard (tablero de tareas de scrum) y el gráfico burndown. Cabe mencionar que estos artefactos deben ser actualizados diariamente para la toma de decisiones.

La última interrogante permite solucionar los problemas que bloquean el flujo de las actividades. Estos obstáculos se deben informar al Scrum Master para que sean solucionados los más rápido posible.

a. Tablero de tareas

Es una forma sencilla pero eficaz que muestra el estado actual del sprint en una sola mirada. Los estados de tareas que se visualizan en el tablero son los siguientes: pendientes, en curso y terminadas.
Este tablero debe ser actualizado de forma diaria por el equipo de desarrollo.

b. Gráfico burndown

Esta gráfica de burndown es bastante útil para efectuar el seguimiento del progreso de nuestro proyecto software, pero también se utiliza como un indicador para predecir si el equipo completará el trabajo en el tiempo estimado. El gráfico burndown debe estar relacionado con la tabla del sprint backlog y el tablero de tareas de scrum (scrum taskboard) para visualizar la tendencia del gráfico burndown inmediatamente. El gráfico burndown debe mostrar una tendencia más pequeña cada día a lo largo del sprint, puesto que las tareas están siendo implementados por el equipo de desarrollo.

Los miembros del equipo de desarrollo son los responsables de actualizar el tablero de tareas después de cada jornada de trabajo, para tomar decisiones oportunas en base a la tendencia que adopta el gráfico burndown.

En la figura 18 apreciamos la gráfica burndown. En eje vertical (Y), se logra visualizar la estimación total de las tareas en horas de esfuerzo y en eje

horizontal (X), se muestra los días planificados equivalente a un sprint, en este caso un sprint de quince días.

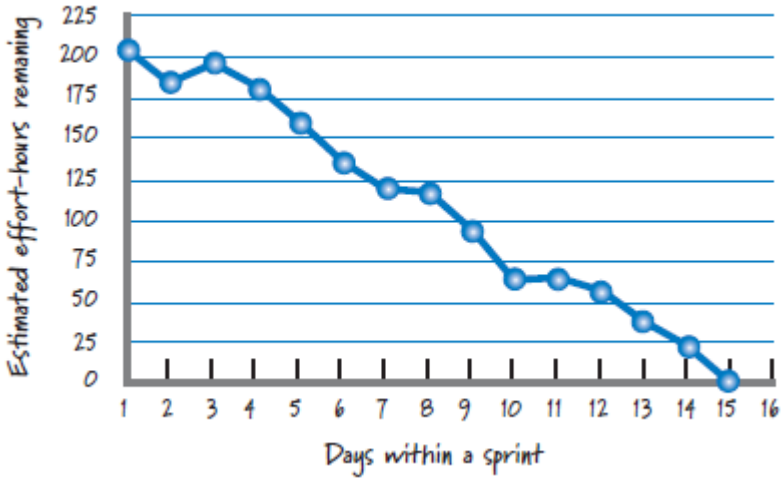


Figura 18. El Gráfico burndown

Fuente: Adaptado de Rubín (2013)

Estándares de codificación

Son un conjunto de reglas y convenciones que deben ser utilizados por el equipo de desarrollo para escribir las instrucciones. Al utilizar estas reglas y convenciones a lo largo de un proyecto, obtenemos los siguientes beneficios: fortalecimiento de la propiedad colectiva del código, reducir los errores al momento de realizar la integración, acrecienta la productividad de los desarrolladores y reduce los costos de mantenimiento.

Los estándares de codificación varían dependiendo del lenguaje de programación, a continuación, se mencionan algunas reglas que se debe utilizar.

a. Comentarios en los programas

"Permiten documentar los programas y mejorar su legibilidad" (Deitel y Deitel, 2016).

//Este método permite sumar dos números

b. Nombre de clases e identificadores

Todos los nombres de las clases deben comenzar con una letra mayúscula y la primera letra de cada palabra en el nombre de la clase debe ir en mayúscula (por ejemplo, PersonalMedico).

c. Identificadores para valores constantes.

Para dar nombre a las constantes utilice únicamente letras mayúsculas; en constantes con más de dos palabras utilice guion bajo.

d. Comentarios en los programas

Los métodos deben ser verbos, con la primera letra del nombre en minúscula, y con la primera letra de cada palabra interna en mayúscula.

Propiedad colectiva del código

Esta práctica autoriza a todos los desarrolladores que participan en el proyecto, acceder al código fuente del sistema y están facultados para corregir posibles errores, independientemente de que hayan trabajado en esta parte del código. A través de esta práctica reduciremos el riesgo de ralentizar el ritmo de desarrollo cuando uno de los miembros de desarrollo abandone el proyecto.

Diseño simple

"Esta práctica permite que un sistema sea fácil de mantener, comprensible y, por tanto, cambiante", en ese sentido enfatizamos que solo debemos implementar funcionalidades que necesita el cliente y no lo que pueda necesitar más adelante. Para lograr un diseño simple y conciso debemos utilizar patrones de diseño de software conocidos, refactorización de código y el desarrollo basado en pruebas.

La Metáfora del sistema

A través de la metáfora, los miembros del equipo deben desarrollar una visión común sobre cómo funciona el sistema, puesto que la metáfora es una descripción simple relacionado al funcionamiento del sistema.

Las Pruebas ágiles

Es una práctica de pruebas de software que obedece los principios del desarrollo ágil de software y se realizar en todo el proceso de desarrollo. Las

pruebas del software no se consideran una fase aislada, sino como parte integral del desarrollo de software. Las pruebas ágiles permiten mejorar la calidad del software, reutilización del código y la escalabilidad de la aplicación a largo plazo.

Las pruebas ágiles, incorpora una serie de prácticas tales como: Las pruebas unitarias y las pruebas de aceptación.

a) Las Pruebas Unitarias

Todas las pruebas unitarias deben cumplir con las cinco características para ser considerados pruebas de calidad y son los siguientes: rápido, independiente, repetible, auto evaluable y oportuno. En caso de no cumplir con las características mencionadas, no garantiza de que sea una prueba unitaria, aunque se utilice herramientas automatizadas del tipo xUnit para su ejecución.

Los desarrolladores pueden utilizar una de las siguientes herramientas que le ayudarán con las pruebas, teniendo en cuenta el lenguaje de programación que están utilizando:

a. **Junit.** Es un conjunto de plugins para realizar pruebas unitarias de aplicaciones Java.

b. **NUnit.** Permite realizar pruebas unitarias en plataformas .NET.

c. **PHPUnit.** Entorno de pruebas unitarias en el lenguaje de programación PHP.

b) Pruebas de aceptación

Es una de las últimas pruebas que se realiza directamente con el Propietario del Producto. El propietario del producto valida las funcionalidades implementados durante el sprint en base a las condiciones de satisfacción especificados al momento de plantear las historias de usuario. Cuando se determina que el software cumple con los requisitos de negocio estará listo para la presentación en la revisión del sprint.

Integración Continua

"La integración continua verifica la calidad del código continuamente durante todo el proceso, en lugar de dejar todo el control de calidad que se realizará al final de un proyecto".

Para asegurar que todo el código que el equipo está desarrollando realmente funciona, los miembros del equipo realizan la integración del código en un servidor de integración continua (CI) que extraerá el código base de un sistema de control de versiones, para luego ejecutar el script de compilación y todas las pruebas automatizadas. El proceso de compilación puede tener éxito o fallas. Cuando la compilación falla se debe notificar instantáneamente a desarrolladores y gerentes de proyectos para corregir el código de forma inmediata. El seguimiento diario proporciona un indicador de progreso en la creación de valor para el cliente. Este ciclo de retroalimentación rápida también permite a los desarrolladores corregir la compilación cuanto se identifique un problema, puesto que esto resulta más barato de arreglarlo.

La integración continua es muy productiva en el sentido de que aminora los épicos inconvenientes de integración al finalizar un proyecto software y nos ofrece la capacidad de realizar la integración continua de forma permanente, rápida, sencilla y en menor tiempo.

3.6.4.4. Revisión del Sprint

Es una actividad que posee como objetivo la inspección y adaptación de las funcionalidades del producto software que se ha implementado. La revisión del sprint establece una visión transparente del estado vigente del producto software, puesto que es una oportunidad para plantear interrogantes, realizar observaciones o sugerencias y tener discusiones acerca del incremento del producto software.

Los incrementos generados en un sprint constituyen parte de los pequeños lanzamientos.

Pequeños lanzamientos

Un lanzamiento significa que los incrementos son los más cortos posibles, además cada lanzamiento demuestra que tiene valor para el cliente y son

respaldados como productos de calidad. Estos lanzamientos constituyen una oportunidad para ver el progreso de proyecto y las nuevas versiones que surgen.

Participantes

Mediante esta actividad, el equipo Scrum, obtiene realimentación de los interesados del negocio que habitualmente no participan durante la ejecución del sprint. La revisión del sprint es una oportunidad, para visualizar y discutir el trabajo que se ha realizado a lo largo el sprint. Por consiguiente, es necesario la participación de los interesados del negocio previa una invitación del equipo Scrum. Quienes confirman que las funcionalidades fueron implementadas, cabe mencionar que ellos son los que indican la programación de la reunión y precisando la fecha, lugar y el tiempo previsto para desarrollar la reunión.

Dinámica de la reunión para la revisión del sprint

Esta actividad empieza con la participación de un miembro del equipo Scrum, generalmente el propietario del producto, quien explica de forma resumida presenta los objetivos del sprint, los ítems de la pila del producto concluidos a lo largo del sprint, en caso que haya objetivos que no se ha logrado, también debe ser mencionado. Inmediatamente uno o más miembros del equipo Scrum empiezan con la demostración de todas las funcionalidades que se ha construido a lo largo el sprint. Culminado la demostración, el equipo Scrum responde las interrogantes de los participantes, esta información obtenida producto de las interrogantes constituye la retroalimentación sobre cómo adaptar el producto. Finalmente, los participantes logran entender las funcionalidades desarrolladas por el equipo Scrum. En caso de observación de algunas funcionalidades se debe incluir en el próximo sprint.

3.6.4.5. Retrospectiva del Sprint

Esta actividad es un contribuyente trascendental como parte de la mejora continua, que consiste en la inspección y adaptación del proceso que se ha empleado para la construcción del producto software en un sprint.

Durante esta actividad se reúnen todos los miembros del equipo Scrum para examinar, analizar e identificar las formas de mejorar el proceso en el próximo sprint. En esta actividad se debe responder las siguientes interrogantes.

a. ¿Qué funcionó bien?
b. ¿Qué no funcionó bien?
c. ¿Qué debemos comenzar a mejorar o hacer de manera diferente?

En función a las respuestas obtenidas, los miembros del equipo establecen algunos cambios accionables para aplicar en el próximo sprint, y de esta manera alcanzar un proceso más óptimo.

Participantes

En esta actividad solamente participan los miembros del equipo Scrum. Esto comprende a los miembros del equipo de desarrollo, el Scrum Master y el propietario del producto. Estos miembros participan activamente argumentando sus perspectivas para mejorar el proceso de construcción del producto software.

El Scrum Master debe mostrar un nivel de liderazgo efectivo y facilitador con respecto a los demás participantes a lo largo de la retrospectiva del sprint.

Dinámica de la retrospectiva del sprint

Esta actividad inicia cuando el Scrum Master verifica los aspectos importantes del proceso que se ha utilizado para la construcción del producto software durante el sprint actual y recopila los datos objetivos para analizar con los demás miembros del equipo Scrum. El Scrum Master es el responsable de elegir ejercicios dinámicos que podrían ayudar a los participantes a pensar, identificar, recopilar, explorar e interpretar los datos necesarios que se utilizaron a lo largo del sprint para implementar acciones de mejora en el próximo sprint. Cabe mencionar que los datos objetivos son los eventos como los ítems de la pila del producto que se iniciaron, pero no se terminaron, el gráfico burndown que muestra el flujo de trabajo completado, no obstante, los datos subjetivos son las percepciones, sentimientos e ideas de los participantes.

Durante la reunión de retrospectiva del sprint, se debe responder a las siguientes interrogantes:

a. ¿Qué funcionó bien durante este sprint?
b. ¿Qué no funcionó bien durante este sprint?
c. ¿Qué debemos mejorar para el próximo sprint?

Existen muchos ejercicios dinámicos que se utiliza en la retrospectiva del sprint para identificar los datos objetivos como subjetivos. Los ejercicios más utilizados en una retrospectiva de sprint son: “diagrama de eventos” y un “sismógrafo de emociones”.

En el presente estudio planteamos la retrospectiva del sprint en base a dos aspectos importantes; retrospectiva desde la perspectiva de gestión y administración, donde proponemos utilizar dos ejercicios dinámicos: “diagrama de eventos” y “sismógrafo de emociones”; asimismo planteamos la retrospectiva a nivel de desarrollo, mediante la práctica de “refactorización del código”.

A. Retrospectiva desde la perspectiva de gestión y administración

a. Cronología del evento

“Crear una línea de tiempo del evento es una forma simple pero poderosa de generar un artefacto compartido que representa visualmente el flujo de eventos durante un sprint. Un enfoque común es dibujar una línea de tiempo en una pizarra y que los participantes coloquen tarjetas en la línea del tiempo que representa eventos significativos que ocurrieron durante un sprint. Las tarjetas se colocan en la línea del tiempo en orden cronológico” (Rubín, 2013).

b. Sismógrafo de emociones

“El sismógrafo de emociones es un complemento del ejercicio de cronología del evento. Esta es una representación gráfica de los altibajos emocionales del participante en el transcurso del sprint. Creando un sismógrafo de emociones ayuda a compartir el contexto más allá de los datos objetivos (lo que sucedió) para incluir algunos datos subjetivos (Como se sintió el equipo al respecto) Para crear el sismógrafo, se invita a cada participante a dibujar una curva que muestre como se sintió o como era su nivel de energía en el transcurso del sprint” (Rubin, 2013).

B. Retrospectiva a nivel de desarrollo de software

Refactorización del código

Al utilizar esta práctica mejoramos el diseño del código después de que se haya escrito a lo largo de un sprint de quince días. La refactorización ayuda a construir un código mantenible y adaptable al cambio, además se recomienda refactorizar utilizando patrones de diseño conocidos.

3.6.5. RESUMEN DE LAS BUENAS PRÁCTICAS EN EL MODELO DE DESARROLLO DE LA PROGRAMACION EXTREMA SOBRE SCRUM

Las buenas prácticas de XP se han incorporado a las principales actividades del "Modelo de desarrollo de software de la Programación Extrema sobre Scrum". En esta tabla se muestra un resumen de la relación de las prácticas de Scrum y XP en el modelo propuesto.

Tabla 18

Relación de actividades de Scrum con las prácticas de XP en el modelo propuesto

Actividades de Scrum		Prácticas de XP
Planificación del sprint		**Ritmo sostenible.** Una planificación adecuada en función a las estimaciones reales y comparables con la velocidad, permitirá lograr los objetivos sin la necesidad de utilizar horas extras.
Ejecución del sprint	**Scrum diario** (Tablero de tareas, Grafico burndown)	**Diseño simple.** Mantener el diseño de un sistema simple y conciso utilizando las buenas prácticas como patrones de diseño, refactorización y pruebas unitarias. **Programación en parejas.** Fomenta el intercambio de conocimiento, aumenta la calidad del código y reduce los errores. **Propiedad colectiva del código.** Establece que todos los miembros puedan acceder al código fuente y por ende están autorizados a realiza modificaciones en cualquier parte del código del proyecto. Teniendo en cuenta que no falle al momento de realizar la integración. **Desarrollo dirigido por pruebas.** Obliga a escribir mínimo código posible, al escribir pruebas antes de escribir las instrucciones. **Estándares de codificación.** Obliga a los desarrolladores a utilizar las reglas y convenciones para escribir código, está relacionado con la propiedad colectiva del código. **Integración continua.** Impulsa a que los desarrolladores envíen los cambios de forma periódica a un repositorio compartido con un sistema de control de versiones para evitar

Actividades de Scrum	Prácticas de XP
	errores, mejorar la calidad del software y reducir el tiempo que se tarda en validar las nuevas actualizaciones. **Pruebas de aceptación.** Comprobar que las historias de usuario implementados satisfagan las necesidades del cliente en base a las condiciones de satisfacción escritos inicialmente.
Revisión del sprint	**Pequeños lanzamientos.** Entrega valor comercial real en un ciclo muy corto, genera retroalimentación de parte de los clientes, reduce los riesgos en el lanzamiento final.
Retrospectiva del sprint	**Refactorización.** Permite reestructurar el código fuente, alterando la estructura interna, pero sin realizar cambios en la funcionalidad de la aplicación.

Fuente: Elaboración propia

CAPÍTULO IV

ANÁLISIS Y RESULTADOS DE LA INVESTIGACIÓN

4.1. APLICACIÓN DEL "MODELO DE DESARROLLO DE LA PROGRAMACIÓN EXTREMA SOBRE SCRUM"

4.1.1. Fase de preparación (Sprint Cero)

Durante esta fase de preparación, que inicia quince días antes de realizar la planificación del primer Sprint, se ha construido las historias de usuario juntamente con el propietario del producto y además se ha definido las herramientas que se utilizarán, la arquitectura de la aplicación y también se ha entregado el manual de los estándares de desarrollo. En seguida, se muestran las siguientes historias de usuario.

Tabla 19

Historia de Usuario. Acceder a la plataforma

HISTORIA DE USUARIO		
ID: HU-01		**Usuario:** Administrador del sistema
Nombre de la historia: Acceso al sistema (Login)		
Prioridad del Negocio: Alta		**Riesgo en Desarrollo:** Alta
Descripción:	Como usuario del sistema necesito acceder al sistema	
Condiciones de satisfacción:	El usuario y la contraseña debe ser creado por el administrador y posteriormente la contraseña debe ser modificado por el usuario.	

Fuente: Elaboración propia

Tabla 20

Historia de Usuario. Registrar datos del paciente

HISTORIA DE USUARIO	
ID: HU-02	**Usuario:** Personal responsable
Nombre de la historia: Registrar paciente	
Prioridad del Negocio: Alta	**Riesgo en Desarrollo:** Alta
Descripción:	Como personal responsable necesito registrar los datos del paciente en el sistema.
Condiciones de satisfacción:	Los datos obligatorios que se deben registrar son: nombres y apellidos, Documento Nacional de Identidad (DNI), edad, dirección y teléfono.

Fuente: Elaboración propia

Tabla 21

Historia de Usuario. Buscar datos del paciente

HISTORIA DE USUARIO	
ID: HU-03	**Usuario:** Personal responsable
Nombre de la historia: Buscar datos del paciente	
Prioridad del Negocio: Alta	**Riesgo en Desarrollo:** Alta
Descripción:	Como personal responsable necesito buscar los datos del paciente en el sistema.
Condiciones de satisfacción:	La búsqueda de los datos del paciente se realizará utilizando su nombre o apellido.

Fuente: Elaboración propia

Tabla 22

Historia de Usuario. Modificar datos del paciente

HISTORIA DE USUARIO	
ID: HU-04	**Usuario:** Personal responsable
Nombre de la historia: Modificar datos del paciente	
Prioridad de Negocio: Alta	**Riesgo en Desarrollo:** Alta
Descripción:	Como personal responsable quiero modificar los datos del paciente.
Condiciones De Satisfacción:	El sistema debe permitir modificar la dirección del paciente y/o el número de teléfono del paciente.

Fuente: Elaboración propia

Tabla 23

Historia de Usuario. Registrar datos del médico responsable

HISTORIA DE USUARIO	
ID:HU-05	**Usuario:** Personal responsable
Nombre de historia: Registrar médico responsable	
Prioridad de negocio: Alta	**Riesgo en desarrollo:** Alta
Descripción: como personal responsable necesito registrar los datos del médico responsable de la atención al paciente.	
Condiciones de satisfacción: El sistema debe permitir registrar los siguientes datos del personal médico: nombres y apellidos, Documento Nacional de Identidad (DNI) y la especialidad del personal médico.	

Fuente: Elaboración propia

Tabla 24

Historia de Usuario. Buscar datos del médico responsable

HISTORIA DE USUARIO	
ID: HU-06	**Usuario:** Personal responsable
Nombre de la historia: Buscar datos del médico responsable	
Prioridad de Negocio: Alta	**Riesgo en Desarrollo:** Media
Descripción:	Como personal responsable necesito buscar los datos del personal médico en el sistema.
Condiciones de satisfacción:	La búsqueda de los datos del personal médico se realizará filtrando mediante su nombre o apellido.

Fuente: Elaboración propia

Tabla 25

Historia de Usuario. Modificar datos del médico responsable

HISTORIA DE USUARIO	
ID: HU-07	**Usuario:** Personal responsable
Nombre de la historia: Modificar datos del médico responsable	
Prioridad de Negocio: Alta	**Riesgo en Desarrollo:** Media
Descripción:	Como personal médico quiero modificar los datos del médico responsable.
Condiciones De Satisfacción:	El sistema debe permitirá modificar los datos del personal médico.

Fuente: Elaboración propia

Tabla 26

Historia de Usuario. Registrar información de los signos vitales del paciente

HISTORIA DE USUARIO	
Numero: HU-08	**Usuario:** Personal responsable
Nombre de historia: Registrar signos vitales del paciente	
Prioridad del negocio: Alta	**Riesgo en desarrollo**: Alta
Descripción: Como personal responsable necesito registrar los signos vitales de los pacientes para realizar la clasificación del riesgo.	
Condiciones de satisfacción: Para realizar el registro con éxito será necesario que las demás informaciones estén registradas previamente.	

Fuente: Elaboración propia

Tabla 27

Historia de Usuario. Generar reporte de los pacientes atendidos

HISTORIA DE USUARIO	
ID: HU-09	**Usuario**: Personal responsable
Nombre de la historia: Generación de reporte de los pacientes atendidos	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Media
Descripción: Como personal responsable necesito generar reportes de los pacientes atendidos en un intervalo de fechas para remitir al Ministerio de Salud.	
Condiciones de satisfacción: El intervalo de las fechas de inicio y fin serán seleccionados por el personal responsable para generar el reporte correspondiente, de lo contrario, la información se generará por defecto; es decir, del mes actual.	

Fuente: Elaboración propia

4.1.2. Planificación del sprint

Definición de roles del proyecto

En la Tabla N° 28, se muestra los roles asignados a los participantes en el presente proyecto.

Tabla 28
Definición de roles en el proyecto

Roles	Integrante
Scrum Master	Luis Miguel Prado Vásquez
Propietario del producto	Dr. Walter Melgar Salcedo
Equipo de desarrollo	• Juan Quispe Ayala • Carlos Bautista Chávez • Edgar Gómez De la Cruz

Fuente: Elaboración propia

a) Primera parte: Reunión para la estimación del product backlog

Esta actividad consiste en la estimación de los ítems de la pila del producto utilizando la técnica del Planning Poker, durante esta actividad participaron todos los miembros del equipo Scrum y se estableció como objetivo la implementación de nueve historias de usuario tal como se muestra en la Tabla N° 29. Se ha determinado que la duración del sprint será de quince días calendario.

b) Segunda parte: Reunión para la estimación del sprint backlog

Después de realizar la estimación de los ítems de la pila del producto, se ha procedido a descomponer los ítems en un conjunto de tareas a nivel de desarrollo y se realizó la estimación correspondiente en horas de esfuerzo con la participación activa del equipo de desarrollo. Cabe mencionar que en esta segunda parte de la reunión participaron el Scrum Master y los miembros del equipo de desarrollo, puesto que no es indispensable la participación del propietario del producto. La estimación de estas tareas se logra visualizar en la Tabla N° 30.

Tabla 29

Estado inicial del backlog del producto en el sistema de Emergencias Obstétricas

Identificador (ID)	Enunciado de la historia de usuario o ítem del backlog del producto	Prioridad	Sprint Número	Estimación (Puntos de historia)	Módulos
1	Iniciar sesión	Alta	1	5	Login
2	Registrar paciente nuevo	Alta	1	8	Gestión de pacientes
3	Buscar datos del paciente	Alta	1	3	
4	Modificar datos del paciente	Alta	1	3	
5	Registrar datos del personal medico	Alta	1	8	Gestión de personal médico
6	Buscar datos del médico	Alta	1	3	
7	Modificar datos del médico	Alta	1	3	
8	Registrar los signos vitales	Alta	1	8	Registro de signos vitales
9	Generar reporte de pacientes atendidos	Media	1	5	Reporte

Fuente: Elaboración propia

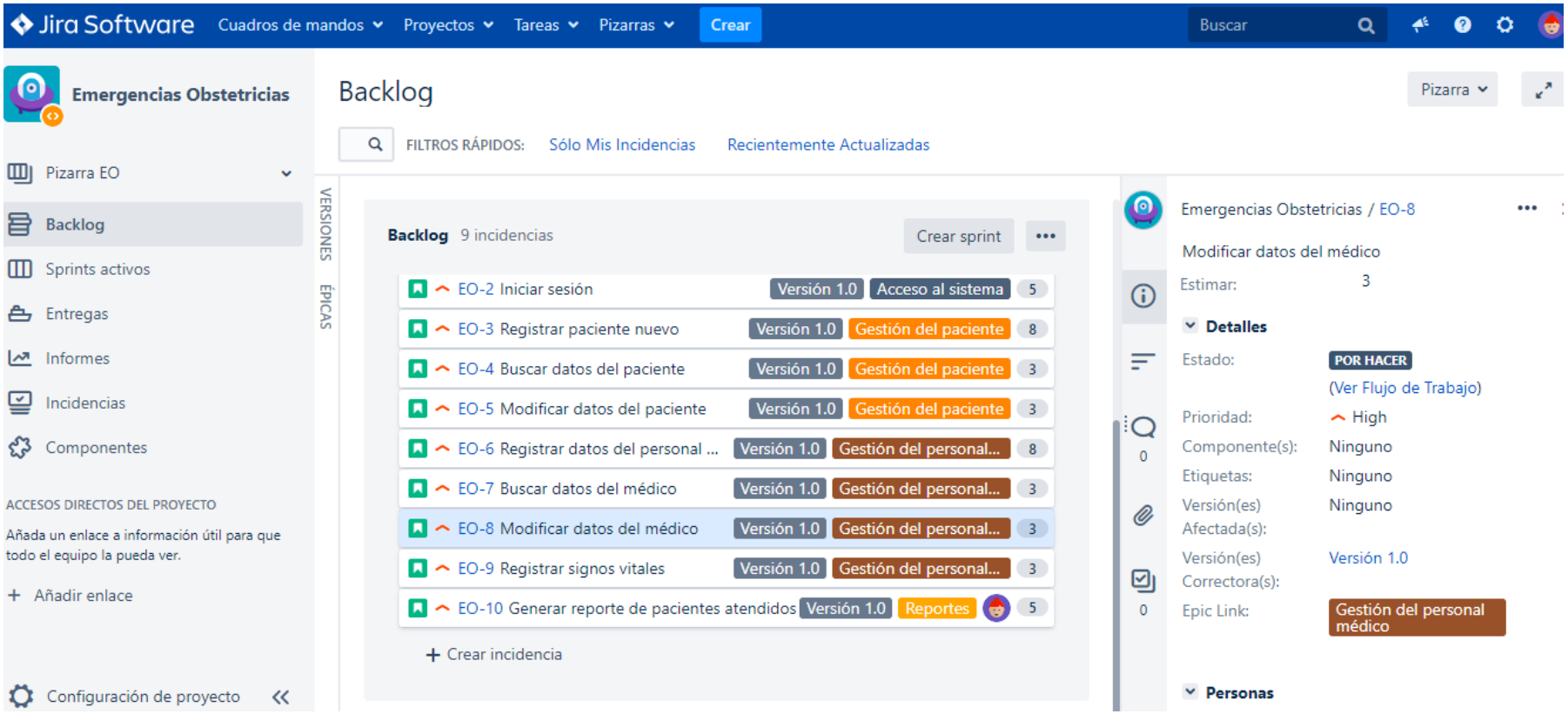


Figura 19. Ítems de la pila del producto en la herramienta Jira

Nota: Los ítems de la pila del producto que se muestran en la figura pertenecen a la versión 1.0 y están agrupados en diferentes épicas.

Fuente: Elaboración propia

Tabla 30

La pila del sprint en el sistema de Emergencias Obstétricas

Identificador del Sprint Backlog	Tareas a nivel de desarrollo	Tipo	Estado	Responsable	Días	D1	D2	D3	D4	D5
					Horas pendientes	55	36	19	12	12
HU-01	Crear la tabla cuenta en la base de datos	Diseño	Pendiente	Juan		2h	0	0	0	0
HU-01	Crear procedimiento para insertar en la tabla cuenta	Programación	Pendiente	Carlos		2h	1h	0	0	0
HU-01	Crear la interfaz de usuario para la tabla usuario	Diseño	Pendiente	Edgar		3h	2h	0	0	0
HU-01	Crear la lógica de la interfaz acceso al sistema en la capa de negocio	Programación	Pendiente	Edgar		5h	2h	0	0	0
HU-02	Crear la tabla paciente	Diseño	En curso	Juan		1h	0	0	0	0
HU-02	Crear la tabla servicio	Diseño	En curso	Juan		1h	0	0	0	0
HU-08	Crear la tabla signos vitales	Diseño	En curso	Juan		1h	0	0	0	0
HU-02	Crear los procedimientos en la base de datos para las consultas, inserciones y modificaciones en la tabla paciente	Programación	Pendiente	Carlos		3h	2h	1h	0	0

Identificador del Sprint Backlog	Tareas a nivel de desarrollo	Tipo	Estado	Responsable	Días	D1	D2	D3	D4	D5
					Horas pendientes	55	36	19	12	12
HU-08	Crear los procedimientos en la base de datos para las inserciones en la tabla signos vitales	Programación	Pendiente	Carlos		2h	2h	1h	0	0
HU-02	Crear el procedimiento para insertar en la tabla servicio	Programación	Pendiente	Edgar		2h	2h	1h	0	0
HU-02	Crear la interfaz de usuario para la gestión de pacientes	Diseño	Pendiente	Carlos		3h	2h	1h	0	0
HU-02	Crear la lógica de la interfaz para la gestión de pacientes en la capa de negocio	Programación	Pendiente	Carlos		5h	2h	1h	1h	1h
HU-05	Crear la tabla personal_medico	Diseño	Pendiente	Juan		2h	0	0	0	0
HU-05	Crear la tabla especialidad	Diseño	Pendiente	Juan		1h	0	0	0	0
HU-05	Crear la tabla colegiatura	Diseño	Pendiente	Juan		1h	0	0	0	0
HU-05	Crear los procedimientos en la base de datos para las consultas, inserciones y modificaciones en la tabla personal_medico	Programación	Pendiente	Edgar		2h	2h	1h	0	0

Identificador del Sprint Backlog	Tareas a nivel de desarrollo	Tipo	Estado	Responsable	Días	D1	D2	D3	D4	D5
					Horas pendientes	55	36	19	12	12
HU-05	Crear el procedimiento para insertar en la tabla colegiatura	Programación	Pendiente	Edgar		3h	3h	0	0	0
HU-05	Crear la interfaz de usuario para la gestión del personal medico	Diseño	Pendiente	Carlos		3h	3h	1h	1h	1h
HU-05	Implementar la lógica de la interfaz para la gestión del personal médico en la capa de negocio	Programación	Pendiente	Carlos		4h	4h	4h	3h	3h
HU-08	Crear la tabla signos_vitales	Diseño	Pendiente	Juan		1h	1h	0	0	0
HU-08	Crear el procedimiento para insertar en la tabla signos_vitales	Programación	Pendiente	Carlos		1h	1h	1h	0	0
HU-08	Diseñar la interfaz para el registro de signos vitales	Diseño	Pendiente	Edgar		2h	2h	2h	2h	2h
HU-08	Implementar la lógica de la interfaz para el registro de los signos vitales	Programación	Pendiente	Carlos		2h	2h	2h	2h	2h
HU-09	Diseñar la interfaz de los reportes	Programación	Pendiente	Juan		3h	3h	3h	3h	3h

Fuente: Elaboración propia

4.1.3. Ejecución del sprint

Scrum Diario

Se ha determinado que esta reunión se realice al inicio de la jornada de trabajo en el horario de ocho de la mañana por un lapso no mayor a 15 minutos todos los días de la semana. Durante esta reunión los miembros del equipo de desarrollo responden a los siguientes interrogantes formulados ¿Qué terminé ayer? ¿Qué es lo que voy a completar hoy? ¿Qué impedimentos u obstáculos (si los hay) estoy enfrentando actualmente? Las dos primeras interrogantes son respondidas rapidamente haciendo uso del tablero de tareas y el gráfico burndown, mientras que la respuesta a la última interrogante es transmitida al Scrum Master para que pueda resolver de forma inmediata, porque son obstáculos que no permite avanzar con las actividades programadas y por ende no se podrán lograr los objetivos en la fecha prevista.

a) Tablero de tareas

"En el ambiente de trabajo del equipo de desarrollo siempre debe estar presente el tablero de tareas como un espacio de trabajo informativo". Para el presente proyecto se ha empleado la herramienta Jira de Atlassian, esta herramienta nos ha permitido gestionar adecuadamente los ítems de la pila del producto y sus respectivas tareas, mediante esta herramienta se logra visualizar el estado evolutivo del sprint, y además nos permite actualizar de forma amigable los ítems del backlog del producto que a su vez está relacionado a las tareas a nivel de desarrollo. En la siguiente Figura N° 20 apreciamos los siguientes estados: por hacer, en progreso y hecho.

b) Gráfico de avance o burndown

"Muestra el estado de avance del proyecto y también permite revisar las posibles desviaciones en la estimación realizada". Para el presente proyecto se ha utilizado la herramienta Jira, como se ha mencionado teóricamente este gráfico de burndown debe estar vinculado con el tablero de tareas y, en efecto, esta herramienta nos proporciona esa funcionalidad, ya que nos muestra el gráfico de burndown en tiempo real a partir de la información del tablero de tareas, como

apreciamos en la Figura N° 21 en el eje “X” visualizamos el tiempo asignado de quince días para el próximo sprint y en el eje “Y” no muestra la estimación con 45 puntos de historia.

Programación en parejas

Es una de las prácticas de la Programación Extrema que se ha utilizado en el presente proyecto, asignando dos desarrolladores para implementar las tareas en los siguientes casos específicos:

- Cuando se implementa una tarea con un nivel de complejidad mayor.
- Cuando se tuvo un desarrollador con poca experiencia frente a otro con vasta experiencia desarrollando aplicaciones.
- El desarrollador que utiliza los estándares de codificación adecuadamente frente a un desarrollador que busca familiarizarse con el manual de los estándares de codificación.

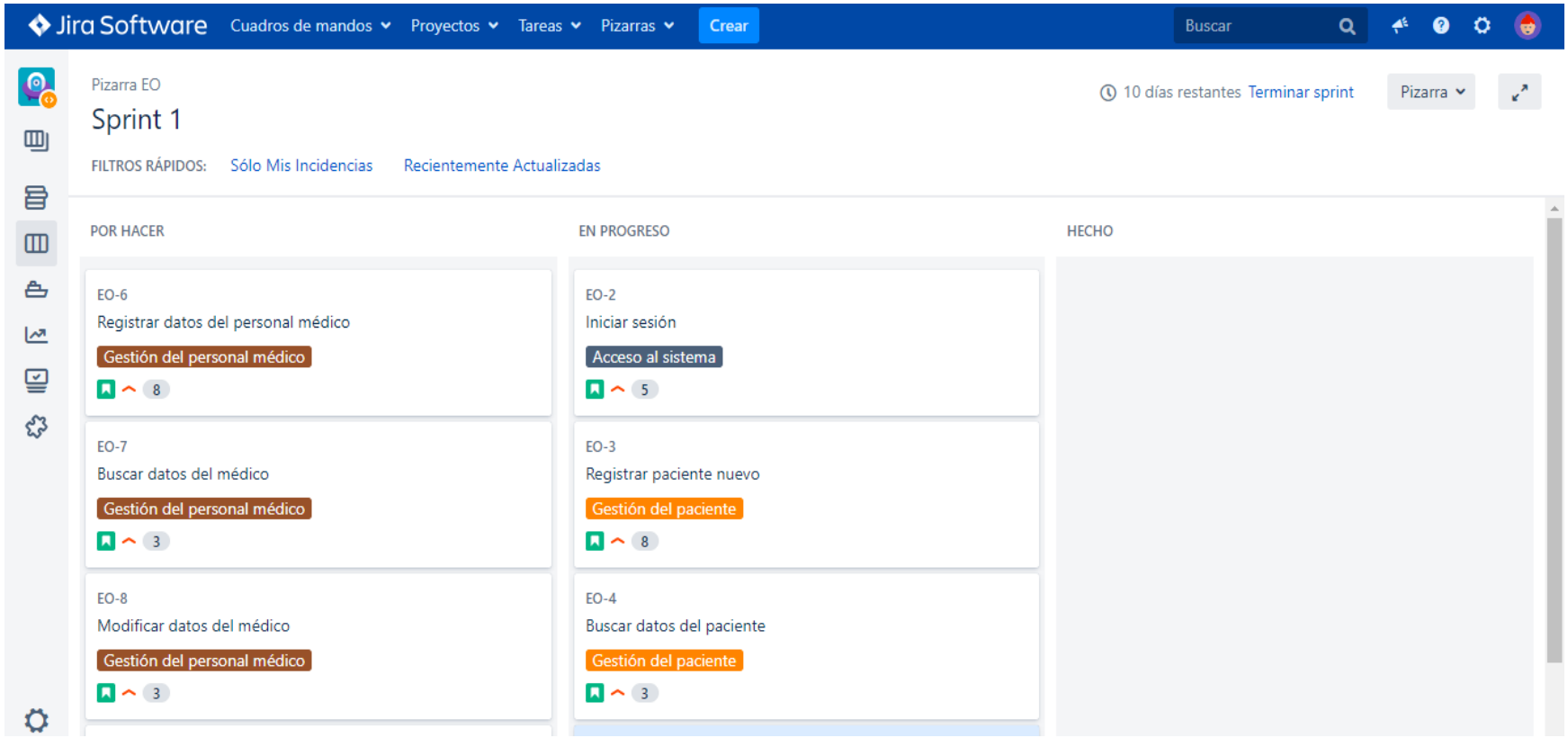


Figura 20. El tablero de tareas en la herramienta Jira

Fuente: Elaboración propia

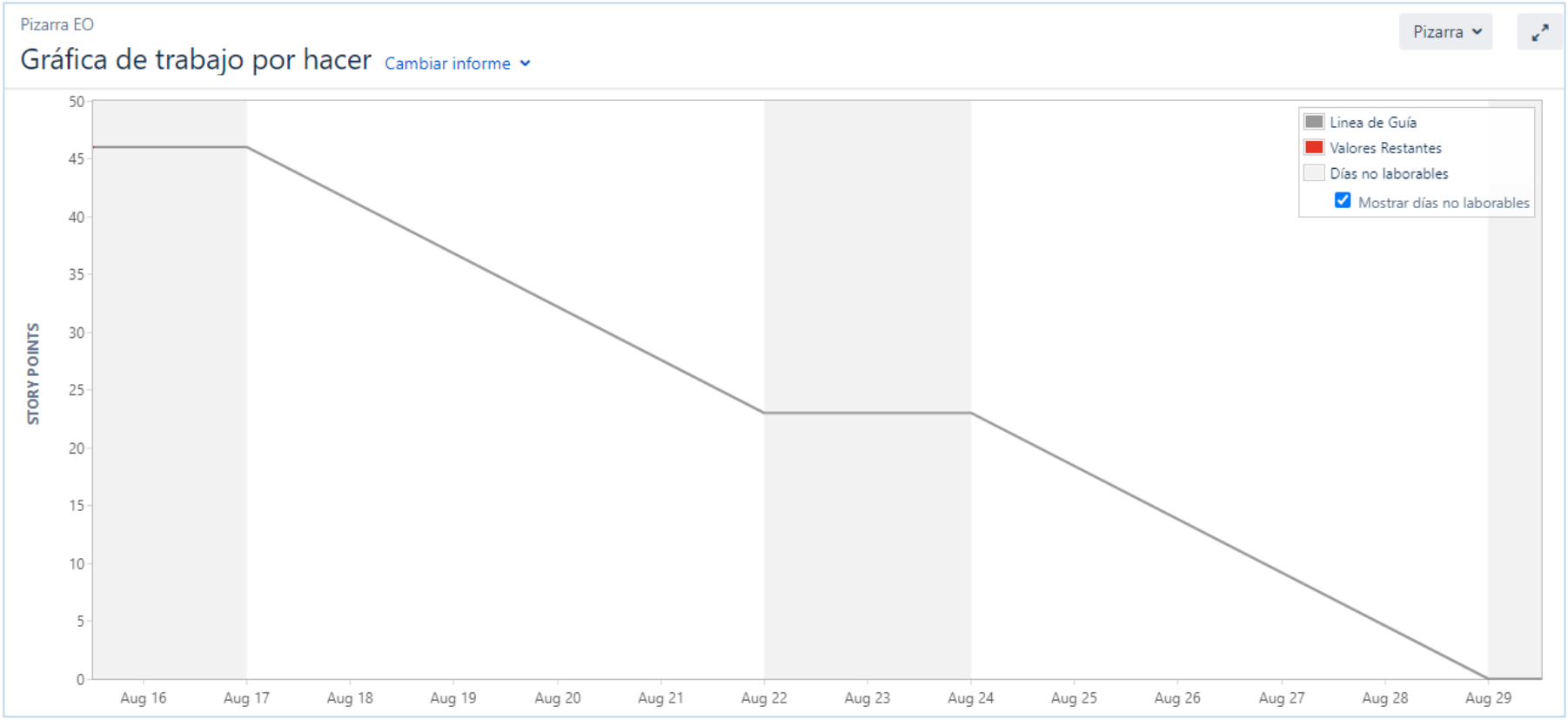


Figura 21. El gráfico burndown al inicio del sprint en Jira

Fuente: Elaboración propia

Figura 22. El gráfico burndown al finalizar el sprint en Jira

Fuente: Elaboración propia

Estándares de codificación

Las reglas y convenciones utilizados por el equipo de desarrollo fueron obtenidos fueron construidos a partir de las buenas prácticas utilizadas en diversos proyectos y en base a la experiencia obtenida por los desarrolladores. En ese sentido, se ha elaborado un manual con los estándares de codificación y fue entregado a los miembros del equipo de desarrollo durante la fase de preparación para que puedan revisar y aplicar durante la ejecución del sprint. En la Figura 23 se muestra el nombre del método en ingles.

```
public void save(BPaciente bPaciente) {
    try {
        clear();
        initExec("public.sp_registrar_paciente(?,?,?,?,?,?,?)");
        setParameterString(1, bPaciente.getNro_historia_clinica());
        setParameterString(2, bPaciente.getNombre_completo());
        setParameterString(3, bPaciente.getDni());
        setParameterString(4, bPaciente.getEdad());
        setParameterInt(5, bPaciente.getNro_partos());
        setParameterString(6, bPaciente.getDireccion());
        setParameterString(7, bPaciente.getTelefono());
        exec();
        conectionCommit();
        bPaciente = null;
    } catch (SQLException e) {
        System.out.println(e);
        bPaciente = null;
        try {
            conectionRollback();
        } catch (SQLException ex) {
            System.out.println(ex);
        }
    } finally {
        bPaciente = null;
    }
}
```

Figura 23. Estándares de codificación en el sistema de emergencias obstétricas

Fuente: Elaboración propia

Pruebas unitarias

Se ha comprobado que cada uno de los métodos implementados en el proyecto funcionan correctamente. Estas pruebas se realizaron de forma manual en el presente sprint, aunque consideramos que lo más adecuado son las pruebas automatizadas a través de herramientas del tipo xUnit. Los métodos que culminaron con éxito las pruebas unitarias se muestran en la Tabla N° 31.

Tabla 31

Pruebas unitarias ejecutadas para verificar los métodos en el sistema de Emergencias Obstétricas

N°	Nombre del método	Resultado esperado	Resultado	Mensaje de error
1	registrar_cuenta	Registrar al usuario en la base de datos	Correcto	Ninguno
2	actualizar_cuenta	Cambiar los datos de conexión del usuario	Correcto	Ninguno
3	registrar_paciente	Registra los datos del paciente en la base de datos	Correcto	Ninguno
4	actualizar_datos_paciente	Permite actualizar los datos del paciente que se encuentra previamente	Correcto	Ninguno
5	buscar_paciente	Muestra los datos del paciente registrado	Correcto	Ninguno
6	registrar_medico	Registra los datos del personal medico	Correcto	Ninguno
7	buscar_datos_medico	Muestra los datos del médico que se encuentra registrado en la base de datos	Correcto	Ninguno

Fuente: Elaboración propia

Tabla 32

Integración de los módulos desarrollados en el sistema de Emergencias Obstétricas

N°	Descripción de los Módulos	Resultado esperado	Estado
1	Iniciar sesión	Permite que el usuario ingrese a la aplicación para realizar sus operaciones y al momento de salir de la aplicación cierra la sesión.	Correcto
2	Gestión del paciente	El usuario que ingresó a la aplicación mediante el login realiza las operaciones de registro, búsqueda y modificación de los datos del paciente. En consecuencia, estos datos son correctamente guardados en la base de datos de la aplicación.	Correcto
3	Gestión del Personal Medico	El usuario que ingresó a la aplicación mediante el login realiza las operaciones de registro, búsqueda y modificación de los datos del médico. Estos datos manipulados son almacenados en la base de datos de la aplicación.	Correcto
4	Registro de los Signos Vitales	El usuario que ingresó a la aplicación mediante el login, registra los signos vitales del paciente. Estos datos ingresados son almacenados en la base de datos de la aplicación.	Correcto
5	Reportes	El usuario previamente logueado genera los reportes a partir de los datos almacenados en la base de datos de la aplicación.	Correcto

Fuente: Elaboración propia

Integración continua

Teniendo en cuenta que esta práctica fomenta la integración en forma permanente en contraposición a la práctica inadecuada que busca realizar la integración al final del proyecto, generando demora, errores de código al integrar y complejidad para entregar el producto software. Se ha realizado la integración de los módulos en forma periódica, utilizando un repositorio central, es así que la integración continua resulta muy beneficiosa para el desarrollo de un proyecto software.

En la Tabla N° 32, visualizamos los módulos integrados a lo largo del desarrollo de la plataforma de emergencias obstétricas del Hospital Regional de Ayacucho.

Pruebas de aceptación o pruebas funcionales

Teniendo conocimiento que estas pruebas permiten verificar que los requisitos del negocio que demanda el cliente se han cumplido satisfactoriamente, se ha procedido a verificar las funcionalidades con la participación del propietario del producto a través del uso de las tarjetas de prueba de aceptación. Cada una de estas tarjetas fueron validadas por el propietario del producto. A continuación, mostramos todas las pruebas de aceptación verificados por el propietario del producto.

Tabla 33

Prueba de aceptación "Registrar datos del paciente"

PRUEBA DE ACEPTACIÓN	
Código: PA-01	**Código Historia de Usuario:** HU-02
Nombre de la prueba: Registrar datos del paciente	
Descripción: Registrar los datos del paciente y guardar en la base de datos.	
Condiciones de ejecución: Ninguno	
Acciones:	

1. Seleccionar la opción “Registrar paciente” del menú Registro. 2. Hacer clic en el botón Nuevo 3. Ingresar los datos del paciente 4. Hacer clic en el botón Guardar
Resultado esperado: La información registrada logra guardarse en la base de datos.
Evaluación de la prueba: Prueba satisfactoria

Fuente: Elaboración propia

Tabla 34

Prueba de aceptación "Buscar datos del paciente en el sistema"

PRUEBA DE ACEPTACIÓN	
Código: PA-02	**Código Historia Usuario:** HU-03
Nombre de la prueba: Buscar datos del paciente en el sistema	
Descripción: El sistema debe buscar información de los pacientes registrados en el sistema.	
Condiciones de ejecución: Los datos consultados en el sistema, previamente deben estar registrados en la base de datos.	
Acciones: 1. Ingresar cualquier filtro de búsqueda (Nro. Historia clínica, nombres). 2. Hacer clic en el icono de búsqueda. 3. obtener la información consultada.	
Resultado esperado: Obtener la información en la sección “Resultados de la búsqueda”	
Evaluación de la prueba: Prueba satisfactoria	

Fuente: Elaboración propia

Tabla 35

Prueba de aceptación "Modificar datos del paciente"

PRUEBA DE ACEPTACIÓN	
Código: PA-03	**Código historia de usuario:** HU-04
Nombre de la prueba: Modificar datos del paciente	
Descripción: Se debe modificar los datos del paciente.	
Condiciones de ejecución: Debe existir información registrado del paciente, para realizar la modificación.	
Acciones: 1. Hacer clic en el botón "Modificar" del formulario registrar paciente 2. Modificar datos necesarios en el formulario. 3. Hacer clic en el botón guardar.	
Resultado esperado: Cuando realizo la búsqueda de los datos del paciente, el sistema me muestra los datos actualizados.	
Evaluación de la prueba: Prueba satisfactoria	

Fuente: Elaboración propia

Tabla 36

Prueba de aceptación "Registrar datos del médico responsable"

PRUEBA DE ACEPTACION	
Código: PA-04	**Código historia de usuario:** HU-05
Nombre de la prueba: Registrar médico responsable	
Descripción: Se debe registrar los datos del médico responsable.	
Condiciones de ejecución: Ninguno	
Acciones:	

1. Seleccionar la opción "Registrar médico responsable" del menú "Registro". 2. Hacer clic en el botón Nuevo. 3. Ingresar la información necesaria. 4. Hacer clic en el botón "Guardar"
Resultado esperado: La información se registra en la base de datos de la aplicación.
Evaluación de la prueba: Prueba satisfactoria

Fuente: Elaboración propia

Tabla 37

Prueba de aceptación "Modificar datos del médico responsable"

PRUEBA DE ACEPTACIÓN	
Código: PA-05	**Código Historia de usuario:** HU-07
Nombre de la prueba: Modificar datos del médico responsable.	
Descripción: Se debe modificar los datos del médico responsable.	
Condiciones de ejecución: debe existir información registrado previamente para su respectiva modificación.	
Acciones: 1. Seleccionar la opción "Registrar médico responsable" del menú "Registro". 2. Hacer clic en el botón "Modificar". 3. Ingresar la información reemplazando datos anteriores. 4. Hacer clic en el botón "Guardar"	
Resultado esperado: Cuando realizo la búsqueda de los datos del personal médico, obtengo los datos actualizados.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 38

Prueba de aceptación "Registrar información de los signos vitales del paciente"

PRUEBA DE ACEPTACION	
Código: PA-06	**Código historia de usuario:** HU-08
Nombre de la prueba: Registrar signos vitales del paciente.	
Descripción: Se debe registrar los signos vitales y los datos antropométricos del paciente.	
Condiciones de ejecución: Ninguno.	
Acciones: 1. Seleccionar la opción "Registrar signos vitales" del menú "Emergencia". 2. Hacer clic en el botón Nuevo. 3. Ingresar los datos en los campos necesarios. 4. Hacer clic en el botón "Guardar"	
Resultado esperado: La información se registra en la base de datos de la aplicación.	
Evaluación de la prueba: Prueba satisfactoria	

Fuente: Elaboración propia

Tabla 39

Prueba de aceptación "Generar reporte de los pacientes atendidos"

PRUEBA DE ACEPTACIÓN	
Código: PA-07	**Código historia de usuario**: HU-09
Nombre de la prueba: Generación de reporte de los pacientes atendidos	
Descripción: Generar el reporte de los pacientes atendidos en la unidad de emergencias obstétricas.	

Condiciones de ejecución: Ninguno.
Acciones: 1. Seleccionar la opción "Reporte de pacientes atendidos" del menú "Reportes". 2. Seleccionar la fecha de inicio y fin. 3. Hacer clic en el botón generar
Resultado esperado: Se logra obtener la lista de los pacientes atendidos en la unidad de emergencias obstétricas.
Evaluación de la prueba: Prueba satisfactoria

Fuente: Elaboración propia

4.1.4. Revisión del sprint

En esta reunión se logró presentar el incremento del producto que se ha construido a lo largo del presente sprint, en esta presentación participaron los responsables de la unidad de emergencias obstétricas y los miembros del equipo Scrum, cabe mencionar que se ha logrado los objetivos planteados durante la planificación del sprint, esto implica que se ha logrado implementar todas las tareas de los ítems del backlog del producto. A continuación, se describe las funcionalidades de la aplicación.

Figura 24. La interfaz principal de la aplicación

En la Figura N° 24, se muestra la pantalla principal de la aplicación, que ha sido implementado en el presente sprint. Esta es una aplicación de escritorio y fue desarrollado con el lenguaje de programación Java. Para visualizar las opciones del menú de la interfaz, el usuario debe acceder a la aplicación mediante un usuario y una contraseña.

En la Figura N° 25, se muestra la interfaz “Registrar Paciente” que también fue implementado en el presente sprint. Esta interfaz permite realizar las siguientes operaciones: registrar datos de los pacientes nuevos, modificar datos de los pacientes existentes, guardar la información registrada y cerrar la interfaz.

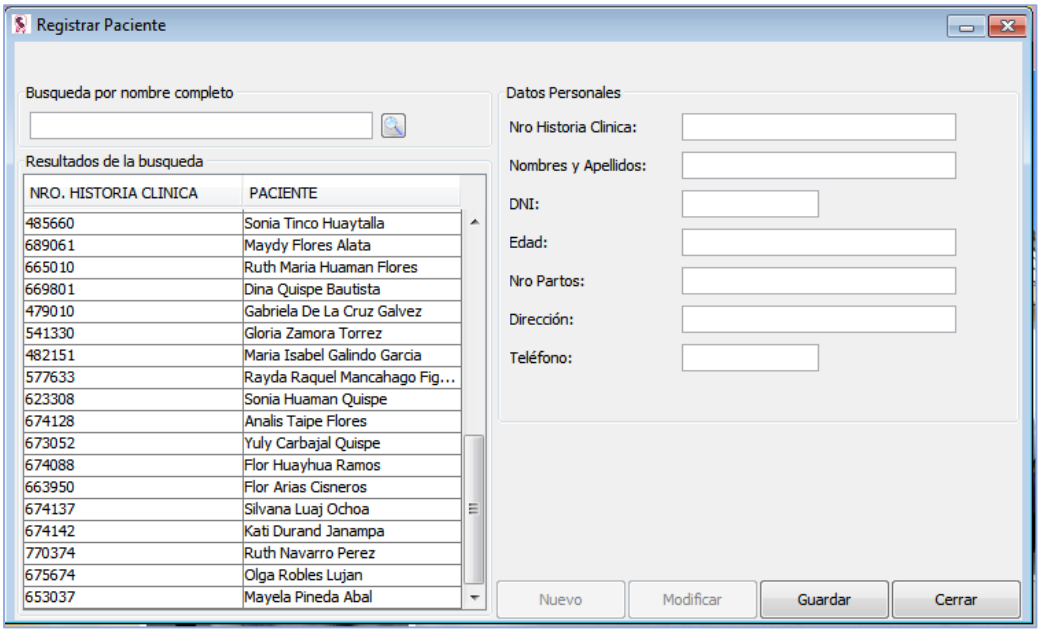


Figura 25. La interfaz registrar paciente

Fuente: Elaboración propia

La Figura N° 26, muestra la interfaz “Registrar Personal Médico” que tambien fue implementado en el presente sprint . Esta interfaz permite realizar las siguientes operaciones: registrar datos de un personal médico nuevo, modificar datos del personal médico, guardar datos y cancelar la información modificada.

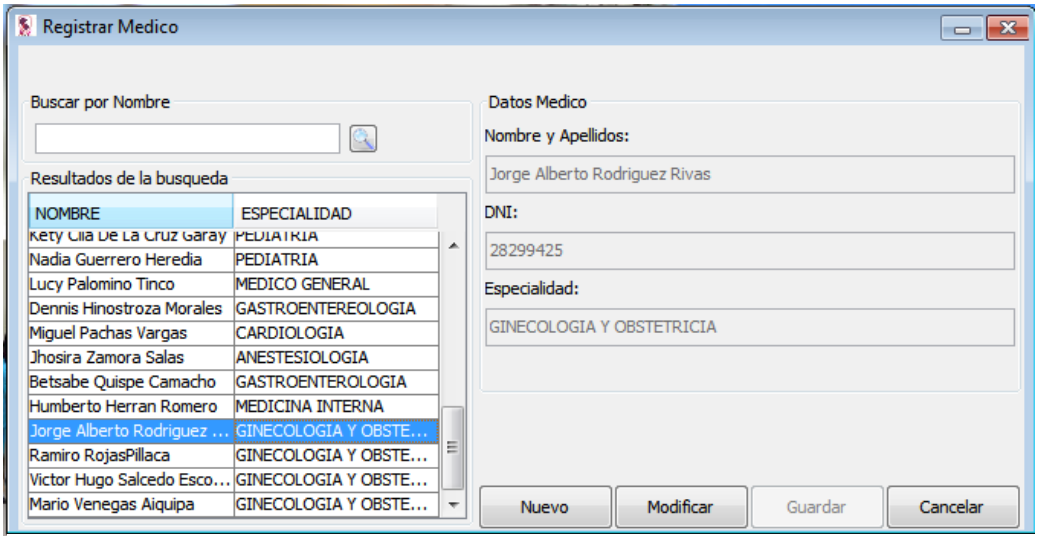


Figura 26. La interfaz registrar médico

Fuente: Elaboración propia

En la Figura N° 27 y 28, se muestra las interfaces relacionados al reporte. Esta interfaz permite generar el reporte de los pacientes atendidos en un rango de fechas; donde el usuario seleccionar el intervalo de fechas y la aplicación muestra los pacientes atendidos en el rango de fechas seleccionado por el cliente. En caso que el usuario no selecciona el rango de fechas, la plataforma mostrará, por defecto, los datos que corresponden al ultimo mes.

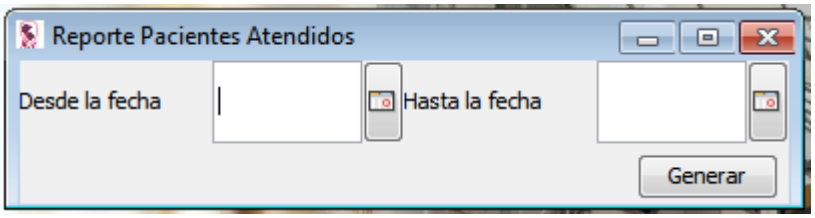


Figura 27. La interfaz del intervalo de fechas para generar reporte

Fuente: Elaboración propia

Figura 28. La interfaz con el reporte de pacientes atendidos

Fuente: Elaboración propia

En la Tabla N° 40, se muestra un resumen del estado final de los ítems de la pila del producto culminados con éxito en el presente sprint.

Tabla 40

Estado final de los ítems de la pila del producto al finalizar el sprint.

Identificador (ID)	Enunciado de la historia	Prioridad	N° de Sprint	Estimación (Puntos de historia)	Estado final
1	Iniciar sesión	Alta	1	5	Terminado
2	Registrar paciente nuevo	Alta	1	8	Terminado
3	Buscar datos del paciente	Alta	1	3	Terminado
4	Modificar datos del paciente	Alta	1	3	Terminado
5	Registrar datos del	Alta	1	8	Terminado

	personal medico				
6	Buscar datos del médico	Alta	1	3	Terminado
7	Modificar datos del médico	Alta	1	3	Terminado
8	Registrar los signos vitales	Alta	1	8	Terminado
9	Generar reporte de pacientes atendidos	Media	2	5	Terminado

Fuente: Elaboración propia

4.1.5. Retrospectiva del sprint

En esta reunión se ha inspeccionado el proceso que se ha utilizado para la construcción del producto software en el presente sprint. En esta reunión participaron únicamente el Scrum Master y el equipo de desarrollo. Después de responder las siguientes interrogantes ¿Qué ha funcionado bien? ¿Qué no ha funcionado bien? ¿Qué debemos mejorar o hacer de manera diferente? Durante la reunión, hemos determinamos realizar cambios que permitan mejorar el proceso, en base a la experiencia obtenida en el presente sprint.

Las acciones que se implementarán abarcan dos aspectos: acciones a nivel de administración y acciones a nivel de desarrollo de código.

a) Acciones a nivel de gestión y administración

Lecciones para mejorar

- Usar herramientas tecnológicas que permita la participación de los interesados del negocio en las reuniones que planifica el Scrum Master.
- Estimar los ítems de la pila del producto de forma realista y objetiva para realizar la implementación a un ritmo sostenible, es decir, sin la necesidad de utilizar horas extras.

- Incorporar como propietario del producto a la persona que realmente conozcan los procesos del negocio.

Lecciones que deben continuar

- Un sprints deben tener una duración de quince días, donde se pueda realizar la refactorización de código y el incremento del producto.
- Se debe continuar realizando las pruebas unitarias, aceptación e integración de forma permanente durante la ejecución del sprint.
- Debemos seguir utilizando la técnica del Planning Poker para realizar la estimación.
- Debemos seguir utilizando la herramienta Jira, para una adecuada administración de las tareas de un proyecto de software ágil.

b) Acciones a nivel de desarrollo

Una de las acciones que se ejecutará utilizando las buenas prácticas de desarrollo es la refactorización del código.

Refactorización

Entendiendo que la refactorización permite reestructurar el código para entregar un producto software libre de errores, flexible, fácil de mantener y escalable en el tiempo. En ese contexto, empleamos la refactorización al finalizar cada sprint. Consideramos necesario el uso de la refactorización en un sprint de dos semanas (15 días), ya que no es posible aplicar esta práctica durante la ejecución del sprint, porque lo más importante es lograr los objetivos planteados durante la planificación del sprint.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

A. De acuerdo al capítulo II, en el numeral 2.2.1, Programación Extrema sobre Scrum, software iterativo e incremental; relacionado con el numeral 2.2.2, gestión de software ágil, planificación del sprint, empleando técnicas e instrumentos sustentados en el capítulo III, sección 3.6.4.2, se ha logrado corroborar que el modelo de desarrollo de software iterativo e incremental de la Programación Extrema sobre Scrum permite la gestión de software ágil a través de la planificación del sprint, cuyos resultados son demostrados en el capítulo IV, aplicado al proyecto del caso práctico.

B. De acuerdo al capítulo II, en el numeral 2.2.1, Programación Extrema sobre Scrum, pruebas unitarias continuas; relacionado con el numeral 2.2.2, gestión de software ágil, ejecución del sprint, usando técnicas e instrumentos sustentados en el capítulo III, sección 3.6.4.3, se ha logrado corroborar que el modelo de desarrollo de software a través de las pruebas unitarias continuas de la Programación Extrema sobre Scrum permite la gestión de software ágil a través de la ejecución del sprint, cuyos resultados son demostrados en el capítulo IV, aplicado al proyecto del caso práctico.

C. De acuerdo al capítulo II, en el numeral 2.2.1, Programación Extrema sobre Scrum, refactorización del código; relacionado con el numeral 2.2.2, gestión de software ágil, retrospectiva del sprint, usando técnicas e instrumentos sustentados en el capítulo III, sección 3.6.4.5, se ha logrado corroborar que el modelo de desarrollo de software a través de la refactorización del código de la Programación Extrema permite la gestión de software ágil a través de la retrospectiva del sprint, cuyos resultados son demostrados en el capítulo IV, aplicado al proyecto del caso práctico.

5.2. RECOMENDACIONES

a. Se recomienda a la comunidad de desarrolladores aplicar el nuevo modelo de desarrollo de software de la Programación Extrema sobre Scrum, así como el uso de la herramienta Jira para administrar de manera eficiente los proyectos de software ágil.

b. Se recomienda desarrollar una herramienta que contenga la propuesta del nuevo modelo de desarrollo.

c. Se recomienda implementar acciones de mejora continua del proceso que se ha utilizado para lograr el incremento del producto software en cada sprint, con la finalidad de obtener un producto software de calidad que satisface las expectativas de nuestros clientes.

d. Se recomienda que el propietario del producto esté plenamente comprometido en el proyecto, participando en las diferentes actividades del equipo Scrum desde que inicia el proyecto hasta concluir la misma. La ausencia del propietario del producto impide lograr los objetivos en la fecha prevista, genera sobrecostos y retraso la entrega del producto software.

REFERENCIA BIBLIOGRÁFICA

Abrahamsson, P., Conboy, K., Morgan, L., Baskerville, R., Fitzgerald, B. y Wang, X. (2008). *Agile Processes in Software Engineering and Extreme Programming*. Germany: Springer-Verlag Berlin Heidelberg.

Alfonzo, P. L., Mariño, S., Godoy, M. V. (2011). *Propuesta Metodológica para la Gestión de Proyecto de Software Ágil basado en la Web*. Multiciencias, 11(4), 395-401.

Arias, A. (2015). *Aprende sobre la Ingeniería de Software*. España, Vigo: IT Campus Academy.

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

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

Bernal, C. A. (2010). *Metodología de la investigación.* Colombia: Pearson Educación.

Blankenship, J., Bussa, M. y Millett, S. (2011). *Pro Agile .Net Development with Scrum.* New York, U.S: Springer Science + Business Media LLC.

Canty, D. (2015). *Agile for Project Managers.* U.S: Taylor & FrancisGroup.

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.

Cohn, M. (2009). *User Stories Applied for Agile Software Development.* U.S: Pearson Education Inc.

Cohn, M. (2006). *Agile Estimating and Planning.* U.S, Massachusetts: Pearson Educations Inc.

Conboy, K. (2009). *Agility from first principles: Reconstructing the concept of agility information systems development. Information Systems Research,*20(3),329 -354.

Dingsoyr, T., Dyba, T., Moe, N. B. (2010). *Agile Software Development*. New York, USA: Springer-Verlag Berlin Heidelberg.

Dooley, J.F. (2017). *Software Development, Design and Coding.* U.S, Illinois.

Feather, M. y Martin, R. C. (2014). *The Art of Unit Testing*. United States of America: Manning Publications Co.

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

Ferreira, R. (2013). *XP Extreme Programming.* Recuperado el 2019 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.

Fowler, M. (2019). *Refactoring Improving the Design of Existing Code.* U.S: Pearson Education.

Freedman, R. (2016). The Agile Consultant. Kansas, USA.

Hanssen, G. K., Stalhane, T., Myklebust, T. (2018). *SafeScrum-Agile Development of Safety-Critical Software.* Switzerland: Springer Nature Switzerland AG.

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

Jurado, C. (2010). *Diseño ágil con TDD.* Córdoba, España: Lulu.com.

Khorikov, V. (2020). *Unit Testing Principles, Practices and Patterns.* United States of America: Manning Publications Co.

Kniberg, H. (2015). *Scrum y XP desde las trincheras*. U.S: C4Media Inc.

Koskela, L. (2008). *Test Driven Practical TDD and Acceptance TDD for Java Developers*. USA: Manning Publications Co.

Kruchten, P. (2007). *Scaling Software Agility*. Vancouver, Canadá.

Koch, A. S. (2005). *Agile Software Development.* U.S, Norwood: Artech House Inc.

Laínez, J. R. (2015). *Desarrollo de Software Ágil Extreme Programming y Scrum*. Recuperado de https://books.google.com.pe/books.

Letelier, P., Penadés, M. C (2006). *Metodologías Agiles para el Desarrollo de Software Extreme Programming.* Valencia, España.

Maximini, D. (2015). *The Scrum Culture Introducing Agile Methods in Organizations*. Switzerland: Springer International Publishing.

Mego, Y. (2020). *Sistema de Información Web bajo la Metodología XP y el Marco de Trabajo Scrum para la Gestión Académica del Instituto Superior Tecnológico Privado Ciro Alegría Morales* (Tesis de pregrado). Universidad Peruana Unión, Perú.

Menzinsky, A., López, G., Palacio, J. (2019). *Scrum Master.* España.

Monroy, S. (2008). *Estadística Descriptiva*. México: Dirección de publicaciones Revillagigedo.

Moreira, M. E. (2017). *The Agile Enterprise*. Massachusetts, USA.

Mushtaq, Z., Rizwan, M. y Jameel, M. R. (2012). *Novel Hybrid Model: Integrating Scrum and XP*.doi: 10.5815/ijitcs.2012.06.06.

Palacio, J. (2008). *Flexibilidad con Scrum.* España: Lubaris Info 4 SL.

Parsons, D. y MacCallum, K. (2019). *Agile and Lean Concepts for Teaching and Learning.* Nueva Zelanda, Auckland: Springer Nature Singapore Pte Ltd.

Pichler, R. (2010). *Agile Product Management with Scrum.* U.S, Massachusetts: Pearson Education Inc.

Pressman, R. (2010). *Ingeniería del Software*. United States, New York: The McGraw-Hill.

Pries, K. H. y Quigley, J. M. (2011). *Scrum Project Management*. U.S: Taylor & Francis Group LLC.

Quezada, N. (2010). Metodología de la Investigación. Lima, Peru: Empresa Editora Macro E.I.R.L.

Qumer, A., & Henderson-Sellers, B. (2008). *A framework to support the evaluation, adoption and improvement of agile methods in practice*. Journal of Systems and Software, 81(11), 1899 – 1919.

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

Sagar, R. (2016). *Mastering Jira 7*. Birmingham, Mumbai: Pack Publishing.

Salazar, J., Tovar, A., Linares, J., Lozano, A. y Valbuena, L. (2018). *Scrum versus XP: similitudes y diferencias*. *TIA,* 6(2), 29-37.

Satpathy, T. (2016). *Scrum Body of Knowledge*. Arizona, USA: SCRUMstudy a brand of VMEdu, Inc.

Schwaber, K. y Sutherland, J. (2017). *The Scrum Guide*. USA: Org and ScrumInc.

Schmidt, C. (2016). *Agile Software Development Teams*. Munchen, Germany: Springer International Publishing Switzerland.

Shore, J. y Warden, S. (2008). *The Art of Agile Development.* United States of America: O'Reilly Media Inc.

Smite, D., Moe, N. B. y Agerfalk, P. J. (2010). *Agility Across Time and Space*. Vancouver, Canada: Springer Verlag Berlin Heidelberg.

Sommerville, I. (2011). *Ingeniería de software*. Madrid, España: Pearson Education Limited.

Stober,T. y Hansmann, U. (2010). *Best Practices for Large Software Development Projects*. Heidelberg, Germany: Springer-Verlag.

Subra, J. P. y Vannieuwenhuyze, A. (2018). *Scrum un Método Ágil para sus Proyectos.* Barcelona, España: Ediciones ENI.

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

Vieyra, V. (2020). *Agile Project: Become Professional in Agile Project Management & Software Development: Agile Software.* USA.

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.

Wake, B (2003). *Extreme Programming Explored.* Massachusetts, U.S: Addison Wesley.

ANEXO A

ANÁLISIS DE CONTENIDO

Tabla 41

Plantilla para recabar información mediante el analisis documental

ANÁLISIS DE CONTENIDO	
Título:	
Autor:	**N° de Página:**
Edición:	**Fecha de publicación:**
Editorial:	**Lugar de publicación:**
Resumen del contenido:	
Nota:	

Fuente: Elaboración propia

ANEXO B

PLANTILLA PARA REGISTRAR LAS HISTORIAS DE USUARIO

Tabla 42

Plantilla para registrar las historias de usuario

HISTORIA DE USUARIO		
Código:		Usuario responsable:
Nombre de la historia de usuario:		
Prioridad del Negocio (Alta, Media, Baja):		Riesgo en Desarrollo (Alta, Media, Baja):
Descripción:	Como [**rol de usuario**], necesito (quiero, puedo) [**acción/funcionalidad**]	
Condiciones de satisfacción:		

Fuente: Elaboración propia

ANEXO C

PLANTILLA DE LA PILA DEL PRODUCTO

Tabla 43

Plantilla para el registro de los ítems de la pila del producto

Identificador (Código)	Enunciado de la historia de usuario	Prioridad (Alta, Media, Baja)	Sprint N°	Estimación (Puntos de historia o días ideales)	Módulo relacionado

Nota: Elaboración propia

ANEXO D

PLANTILLA PARA LA PILA DEL SPRINT

Tabla 44

Plantilla para registrar las tareas de la pila del sprint

Identificador del Ítem de la Pila del Producto	Descripción de la tarea	Tipo (Diseño, Desarrollo, pruebas)	Estado (Pendiente, En curso, Terminado)	Responsable	Días	D1	D2	D3	D4
					Horas pendientes				

Nota: Elaboración propia

ANEXO E

PLANTILLA PARA LA PRUEBA DE ACEPTACIÓN

Tabla 45

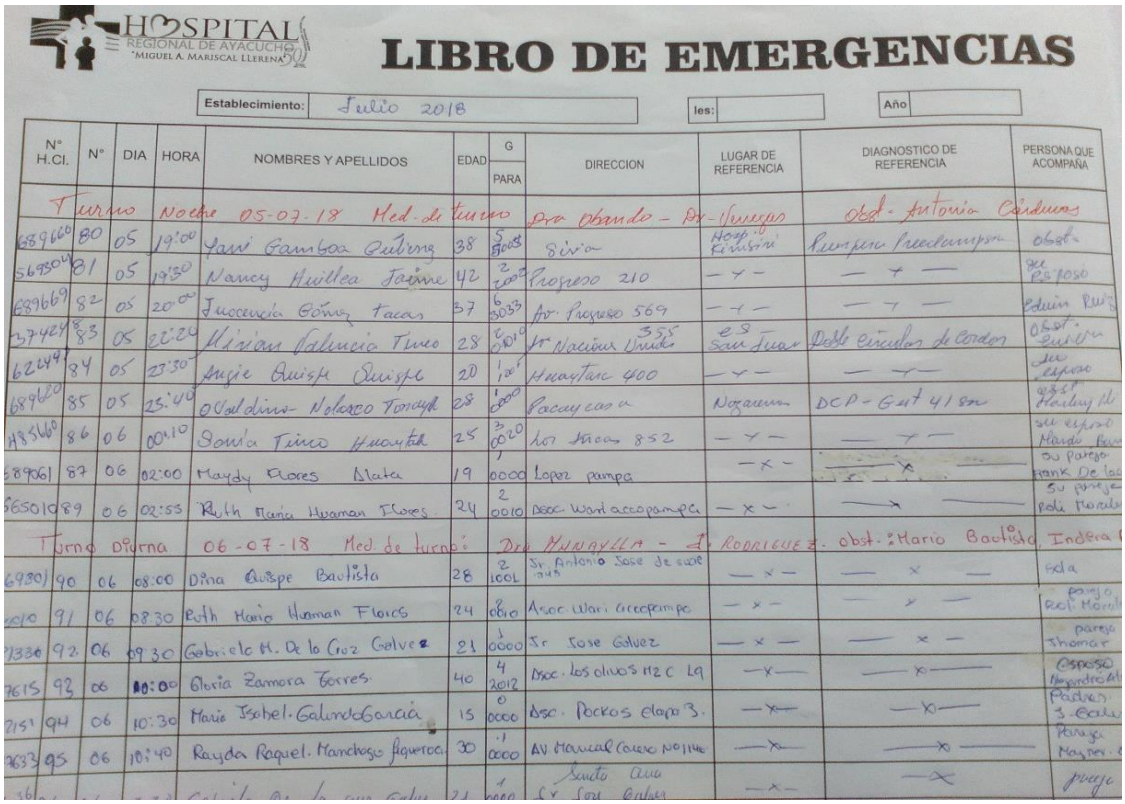
Plantilla de registro de las pruebas de aceptación

PRUEBA DE ACEPTACIÓN	
Código:	**Código Historia de usuario:**
Nombre de la prueba:	
Descripción:	
Condiciones de ejecución:	
Acciones:	
Resultado esperado:	
Evaluación de la prueba:	

Fuente: Elaboración propia

ANEXO F

FOTOGRAFÍAS DEL CONTENIDO DEL LIBRO DE REGISTRO EN LA UNIDAD DE EMERGENCIAS OBSTÉTRICAS



HOSPITAL REGIONAL DE AYACUCHO "MIGUEL A. MARISCAL LLERENA"

LIBRO DE EMERGENCIAS

Establecimiento: Julio 2018 | les: | Año

N° H.Cl.	N°	DIA	HORA	NOMBRES Y APELLIDOS	EDAD	G / PARA	DIRECCION	LUGAR DE REFERENCIA	DIAGNOSTICO DE REFERENCIA	PERSONA QUE ACOMPAÑA
Turno	Noche	05-07-18	Med. de turno				Dra Obando - Dr. Urrutia		Obst. Antonia Cárdenas	
689660	80	05	19:00	Yasmi Gamboa Quispe	38	3 0003	Silva	Hosp. Regional	Puerperio Preeclampsia	obste
569804	81	05	17:50	Nancy Huillca Jaime	42	2 2002	Progreso 210	— x —	— x —	su esposo
689669	82	05	20:00	Inocencia Gómez Tacas	57	6 5003	Av. Progreso 569	— x —	— x —	Edwin Ruiz
37429	83	05	22:20	Miriam Palmeria Tinco	28	0 0000	Jr. Naciones Unidas 355	e.s San Juan	Dolor circulos de Cordon	Obst. Suni
62249	84	05	23:30	Angie Quispe Quispe	20	1 0000	Huaytara 400	— x —	— x —	su esposo
689660	85	05	25:40	Valdivia Velazco Yanayk	28	1 0000	Pacuycasa	Nazarenas	DCP - Gest 41 sem	Obst. Harby M
48566	86	06	00:10	Sonia Tinco Huaytalla	25	3 0020	Los Incas 852	— x —	— x —	su esposo Mario Ba
89061	87	06	02:00	Maydy Flores Mata	19	1 0000	Lopez pampa	— x —	— x —	su pareja Frank De la
65010	89	06	02:55	Ruth Maria Huaman Flores	24	2 0010	Asoc. Wari accopampa	— x —	— x —	su pareja Pedr Manuel
Turno	Diurna	06-07-18	Med. de turno:				Dra MANAYLLA - Dr. RODRIGUEZ.		obst.: Mario Bautista	Indira
69801	90	06	08:00	Dina Quispe Bautista	28	2 1001	Jr. Antonio Jose de sucre 345	— x —	— x —	esta
0/0	91	06	08:30	Ruth Maria Huaman Flores	24	2 0010	Asoc. Wari Accopampa	— x —	— x —	pareja Rol Manuel
71330	92	06	09:30	Gabriela M. De la Cruz Galvez	21	1 0000	Jr. Jose Galvez	— x —	— x —	pareja Shomar
7615	93	06	10:00	Gloria Zamora Torres	40	4 2012	Asoc. los olivos M2 C L9	— x —	— x —	Esposo Alejandro M
2151	94	06	10:30	Maria Isabel. Galindo García	15	0 0000	Asoc. Pachos etapa 3.	— x —	— x —	Padres 3. Gala
2633	95	06	10:40	Rayda Raquel. Manchego Figueroa	30	1 0000	Av. Mariscal Caceres N°1146	— x —	— x —	Pareja Mayori
							Santa Ana Cr. San Carlos	— x —	— x —	pareja

Figura N° F.1. Contenido de la primera sección del libro de emergencias obstétricas.

Figura N° F.2. Contenido de la segunda sección del libro de emergencias obstétricas.

ACTA DE SUSTENTACIÓN DE TESIS N° 053-2020-FIMGC

En la ciudad de Ayacucho, en cumplimiento a la **Resolución Decanal N° 328-2020-FIMGC-D,** siendo los veinticuatro días del mes de noviembre 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, **Dr. Ing. Manuel Avelino LAGOS BARZOLA,** Jurado el **Mg. Ing. Eloy VILA HUAMÁN,** Jurado – Asesor el **Mg. Ing. Hubner JANAMPA PATILLA,** y Secretario del proceso **Mg. Ing. Christian LEZAMA CUELLAR,** con el objetivo de recepcionar la sustentación de la tesis denominada **"MODELO DE DESARROLLO DE SOFTWARE DE LA PROGRAMACIÓN EXTREMA SOBRE SCRUM PARA GESTIÓN DE SOFTWARE ÁGIL, 2019",** sustentado por el Bach. **Luis Miguel PRADO VÁSQUEZ,** bachiller **en Ingeniería de Sistemas.**

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 **16 (dieciséis).**

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

Dr. Ing. Manuel Avelino LAGOS BARZOLA
Presidente

Mg. Ing. Eloy VILA HUAMÁN
Jurado

Mg. Ing. Hubner JANAMPA PATILLA
Jurado – Asesor

Mg. Ing. Christian LEZAMA CUELLAR
Secretario del Proceso

c.c.:
Bach. Luis Miguel PRADO VÁSQUEZ
Jurados (4)
Archivo

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 Sr./Srta.

Apellidos y Nombres	:	PRADO VÁSQUEZ, Luis Miguel
Escuela Profesional	**:**	INGENIERÍA DE SISTEMAS
Título de la Tesis	**:**	"MODELO DE DESARROLLO DE SOFTWARE DE LA PROGRAMACIÓN EXTREMA SOBRE SCRUM PARA GESTIÓN DE SOFTWARE ÁGIL, 2019"
Evaluación de la Originalidad	**:**	**26** % Índice 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, 09 de agosto del 2021

Firmado digitalmente por LEZAMA CUELLAR CHRISTIAN

Mg. Ing. Christian LEZAMA CUELLAR
Verificador de Originalidad de Trabajos de Tesis de Pregrado de la FIMGC

Numero de constancia: **100**-2021-FIMGC.

(X) Con depósito para Sustentación y Tramite de Titulo

“MODELO DE DESARROLLO DE SOFTWARE DE LA PROGRAMACIÓN EXTREMA SOBRE SCRUM PARA GESTIÓN DE SOFTWARE ÁGIL, 2019”

por Luis Miguel Prado Vásquez

Fecha de entrega: 07-ago-2021 11:47a.m. (UTC-0500)
Identificador de la entrega: 1628780173
Nombre del archivo: tesis_PRADO_V_SQUEZ,_Luis_Miguel.pdf (3.16M)
Total de palabras: 24207
Total de caracteres: 137904

“MODELO DE DESARROLLO DE SOFTWARE DE LA PROGRAMACIÓN EXTREMA SOBRE SCRUM PARA GESTIÓN DE SOFTWARE ÁGIL, 2019”

INFORME DE ORIGINALIDAD

26%	15%	1%	22%
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	16%
2	www.risti.xyz Fuente de Internet	1%
3	www.slideshare.net Fuente de Internet	1%
4	docplayer.es Fuente de Internet	<1%
5	Submitted to Universidad Estatal a Distancia Trabajo del estudiante	<1%
6	gitlab.citius.usc.es Fuente de Internet	<1%
7	111learn.com Fuente de Internet	<1%
8	repositorio.utn.edu.ec Fuente de Internet	<1%

9	Submitted to CONACYT Trabajo del estudiante	<1 %
10	repositorio.ucv.edu.pe Fuente de Internet	<1 %
11	repositorio.unsch.edu.pe Fuente de Internet	<1 %
12	Submitted to Universidad Cesar Vallejo Trabajo del estudiante	<1 %
13	hdl.handle.net Fuente de Internet	<1 %
14	repositorio.uigv.edu.pe Fuente de Internet	<1 %
15	repository.unad.edu.co Fuente de Internet	<1 %
16	5d6199f9c934a.site123.me Fuente de Internet	<1 %
17	es.slideshare.net Fuente de Internet	<1 %
18	publicaciones.urbe.edu Fuente de Internet	<1 %
19	vsip.info Fuente de Internet	<1 %
20	ri.uaemex.mx Fuente de Internet	<1 %

21	Submitted to Universidad Pontificia Bolivariana Trabajo del estudiante	<1 %
22	Submitted to UNAPEC Trabajo del estudiante	<1 %
23	documentop.com Fuente de Internet	<1 %
24	aws.amazon.com Fuente de Internet	<1 %
25	repositorio.unap.edu.pe Fuente de Internet	<1 %
26	Submitted to Universidad Autónoma de Aguascalientes Trabajo del estudiante	<1 %
27	Submitted to Instituto Madrileno de Formacion Trabajo del estudiante	<1 %
28	Submitted to Universidad Internacional de la Rioja Trabajo del estudiante	<1 %
29	repositorioacademico.upc.edu.pe Fuente de Internet	<1 %
30	universidadalnus.com Fuente de Internet	<1 %
31	es.scribd.com	

Fuente de Internet

<1 %

32 Submitted to Universidad Catolica de Avila
Trabajo del estudiante

<1 %

33 Submitted to Universidad Nacional de Colombia
Trabajo del estudiante

<1 %

34 Submitted to Universidad Cientifica del Sur
Trabajo del estudiante

<1 %

Excluir citas	Activo	Excluir coincidencias	< 30 words
Excluir bibliografía	Activo		