DEFINICIÓN DE UN PROCESO PARA EL DESARROLLO DE PROYECTOS DE SOFTWARE EN MODALIDAD UNIPERSONAL COMBINANDO ISO/IEC 29110:2014 Y PROCESOS ÁGILES

PEDRO DAVID ERAZO ACOSTA

UNIVERSIDAD AUTÓNOMA DE MANIZALES

FACULTAD DE INGENIERÍA

MAESTRÍA EN GESTIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE

GRUPO DE INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE

MANIZALES, COLOMBIA

2018

DEFINICIÓN DE UN PROCESO PARA EL DESARROLLO DE PROYECTOS DE SOFTWARE EN MODALIDAD UNIPERSONAL COMBINANDO ISO/IEC 29110:2014 Y PROCESOS ÁGILES

PEDRO DAVID ERAZO ACOSTA

TUTOR

MSC. SANDRA VICTORIA HURTADO GIL

UNIVERSIDAD AUTÓNOMA DE MANIZALES

FACULTAD DE INGENIERÍA

MAESTRÍA EN GESTIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE

GRUPO DE INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE

MANIZALES, COLOMBIA

2018

DEDICATORIA

Dedico este nuevo logro personal principalmente a mi familia, empezando por mis padres, Pedro y Gloria, quienes me han apoyado durante cada etapa de mi vida, y a quienes debo cada una de mis metas alcanzadas; a mis hermanas, de quienes siempre he recibido el apoyo, el buen ejemplo, alegrías y buenos momentos que nos mantienen unidos como una gran familia.

De igual manera a mi esposa Paulita, quien representa para mí, un ejemplo de empuje, perseverancia, responsabilidad y superación, y su amor, compañía y apoyo hacen que cada día sea más fácil superar nuevos retos y disfrutar de la vida con alegría y momentos felices.

Pedro David.

AGRADECIMIENTOS

Agradezco infinitamente a mis padres Pedro y Gloria, por cada minuto que han dedicado y por cada esfuerzo que han realizado, para formar y educar una familia a través del buen ejemplo, con acciones, actitudes y enseñanzas que definen la calidad de personas que son y el referente para ser un mejor ser humano cada día.

Agradezco también al Coordinador del programa de Maestría en Gestión y desarrollo de proyectos de software, el doctor Mauricio Fernando Alba Casto, por la oportunidad, la paciencia, y la consideración que tuvo desde el inicio de mi proceso educativo, cuando las condiciones no me favorecían. Siempre obtuve una respuesta positiva y alentadora.

Agradecerle inmensamente a la docente M. Sc. Sandra Victoria Hurtado Gil, por todo el apoyo, la disposición y el tiempo dedicado para sacar mi tesis adelante, y que sin su ayuda y asesoría no hubiese sido posible.

De igual manera quiero darles las gracias a los docentes, quienes durante este tiempo nos acompañaron y compartieron su conocimiento y experiencia, de manera responsable y amena, motivando a sus estudiantes en cada una de las sesiones de trabajo, y que, gracias a ese esfuerzo, fue posible culminar esta etapa de manera satisfactoria.

Por último, quiero agradecer a mis amigos y compañeros de estudio el Ingeniero Carlos Arturo Blandón y el ingeniero Arley Fernando Gallego, por su compañerismo, su buena energía, por compartir conocimiento, por la ayuda, la colaboración y la camaradería. Con esta amistad, este proceso fue aún más agradable y enriquecedor.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE GRÁFICAS

RESUMEN

La globalización ha impactado directamente en todos los sectores industriales, obligando a las organizaciones a modernizar la forma de obtención, almacenamiento y acceso a información en los procesos administrativos, operativos y estratégicos, buscando constantemente la competitividad a partir de la toma de decisiones estratégicas oportunas basadas en la inmediatez, ubicuidad y accesibilidad de la información. En este contexto los desarrolladores de software deben generar productos que cumplan con los requerimientos de las organizaciones, procurando entregas en el menor tiempo posible a la vez, que sean diseñados y desarrollados con una visión de calidad, escalabilidad y crecimiento de la organización o sector para el cual está concebido el desarrollo. En consecuencia, se requiere que los desarrolladores posean mayor control sobre sus recursos, tiempo de desarrollo, costo y calidad del producto.

Las posibilidades de contratación para actividades de desarrollo de software son amplias, desde casas de software, cuyas actividades se realizan cumpliendo procesos de desarrollo rigurosamente, hasta desarrolladores de software que llevan a cabo desarrollos unipersonales presencial o en modalidad de teletrabajo, que por lo general no adoptan protocolos que garanticen la calidad de los productos generados en sus diferentes etapas, así como el cumplimiento en las entregas planificadas.

El presente documento propone la definición de un proceso para el desarrollo de software en modalidad unipersonal, que constituya en sí mismo un proceso de desarrollo ágil que concibe desde sus fases los requisitos establecidos en el estándar ISO/IEC 29110:2014, brindando una oportunidad de fortalecimiento a la actividad de desarrollo de software unipersonal.

Palabras claves: Software, Calidad de Software, Sistemas de Software, Ingeniería de Software, Desarrollo ágil de software

ABSTRACT

Globalization has directly impacted all industrial sectors, forcing organizations to modernize the way of obtaining, storing and accessing information in administrative, operational and strategic processes, constantly seeking competitiveness based on timely strategic decisions based on in the immediacy, ubiquity and accessibility of information. In this context software developers must generate products that meet the requirements of organizations, providing deliveries in the shortest possible time at a time, which are designed and developed with a vision of quality, scalability and growth of the organization or sector for which is conceived the development. Consequently, developers are required to have greater control over their resources, development time, cost and product quality.

The possibilities of contracting for software development activities are wide, from software houses, whose activities are carried out by carrying out development processes rigorously, to software developers who carry out one-person developments in person or in telework mode, which generally does not they adopt protocols that guarantee the quality of the products generated in their different stages, as well as the fulfillment in the planned deliveries.

This document proposes the definition of a process for the development of software in unipersonal mode, which constitutes in itself an agile development process that conceives from its phases the requirements established in the ISO / IEC 29110: 2014 standard, providing an opportunity to strengthening the single-software development activity.

Keywords: Software, Software Quality, Software Systems, Software Engineering, Agile Software Development

1 INTRODUCCIÓN

En los últimos años las tecnologías de información han impulsado un proceso de globalización de gran escala que ha impactado directamente en todos los sectores industriales; bien lo exponía James H. Mittelman, profesor de relaciones internacionales e investigador en economía y política internacional, al considerar que:

La globalización es una fusión de procesos transnacionales y estructuras domésticas que permiten que la economía, la política, la cultura y la ideología de un país penetre en otro. La globalización es inducida por el mercado, no es un proceso guiado por la política. (Mittelman, 1996, pág. 3)

Como consecuencia directa de dicho proceso de globalización, la tecnología se ve avocada a ser más eficiente, segura, confiable y a brindar la sensación de ubicuidad, requiriendo desarrollo de software de calidad que cumpla con los requisitos del mercado, por tanto, los desarrolladores requieren más control sobre recursos, tiempo, costo y calidad del producto.

Dichos procesos de desarrollo de software generan innovación no solo en aplicaciones de escritorio sino también en la internet de las cosas (IoT) y la computación en la nube, las cuales causarán "una revolución en los procesos de producción y mejoran los estándares de vida, sobre todo en los países en desarrollo" (Organización de las Naciones Unidas para el Desarrollo Industrial (ONUDI), 2015, pág. 9).

Actualmente en Colombia, las empresas dedicadas al desarrollo de software tienen niveles de inversión bajos en actividades de I+D+i, lo que "no permite que se generen impactos significativos en los niveles de productividad y competitividad, claves para el desarrollo de la industria TI en el país", (Ministerio de Tecnologías de la Información y Comunicación, 2015), sin contar con las MiPymes de software que realizan desarrollos pequeños, así como aquellos profesionales que realizan estas actividades de manera unipersonal o por teletrabajo cuya adopción de procesos de desarrollo formales es aún más precaria.

Lo anterior aunado a la falta de adopción de estándares internacionales para el desarrollo adecuado de proyectos de software en conjunción con la formación en procesos ágiles representan una gran debilidad en esta industria, bien lo decía (Laporte, 2016)

A medida que la calidad del software se convierta cada vez más en un tema de preocupación, los enfoques del proceso maduran y ganan la confianza en empresas, el uso de las normas ISO se extiende a organizaciones de todos los tamaños. p. 3

La presente tesis propone la definición de un proceso para el desarrollo de software en modalidad unipersonal que combine los requisitos establecidos en el estándar ISO/IEC 29110:2014, (ICONTEC, 2014) con los procesos de desarrollo ágil, como mecanismo que permita fortalecer la industria del desarrollo de software, teniendo en cuenta que "las pequeñas y medianas empresas constituyen el tipo de organización de negocio dominante en el mundo, contabilizando por encima de 95% y hasta 99% del total de negocios dependiendo del país" (ICONTEC, 2014).

Para lograr el objetivo propuesto se llevará a cabo un análisis comparativo de algunos procesos ágiles, una revisión de la NTC ISO/IEC 29110:2014, (ICONTEC, 2014), complementando la información con las opiniones de personas involucradas activamente en el desarrollo de software, concretando elementos que conduzcan a un proceso que garantice la calidad y el cumplimiento de objetivos, finalizando con la generación del proceso definido modelado con EPF-Composer para facilitar su distribución y socialización.

Se espera que la definición de este proceso constituya una herramienta de fácil aplicación para las Pymes y desarrolladores independientes, fortaleciendo la competitividad de esta actividad económica en el país.

2 ANTECEDENTES

(Mogollón Afanador & Estaban Villamizar, 2011), realizan un estudio de la evolución de los procesos de desarrollo de software y plantean la imperiosa necesidad de una adaptación de los procesos de desarrollo grupales a un proceso individual, lo que revierte importancia por los elementos concluidos en la investigación en términos de la necesidad de generar un proceso de desarrollo unipersonal.

(Garces Guayta & Egas, 2012), presentan un resumen evolutivo de los procesos de desarrollo de software, haciendo hincapié en los riesgos, resaltando la importancia de la minimización de los errores, en virtud a que los más perjudiciales se producen en las primeras etapas del desarrollo; de igual manera presentan en su documento una visión general de los procesos para ser implementados por equipos de desarrollo, lo que es pertinente para poder involucrar elementos de gestión de riesgos en el proceso que se pretende definir.

(Colombani, Pérez, & Pacífico, 2016), realizan un análisis de las características, bondades y deficiencias de los diferentes procesos de software, buscando identificar la más adecuada para las pequeñas y medianas empresas que cuentan con equipos de desarrollo de software con cantidad de integrantes limitados, cuyos instrumentos y análisis pueden ser utilizados para contrastar los procesos ágiles y la norma ISO/IEC 29110:2014.

(Britto Montoya, 2014), evalúa las características del proceso de desarrollo de software adelantado en la Caja de Compensación Familiar de Risaralda, comparándolo con las prácticas ágiles. Plantea el diseño de un nuevo proceso el cual fue evaluado posteriormente con respecto de CMMI, obteniendo una matriz que permite evaluar cualquier proceso de desarrollo. El documento reviste importancia desde el proceso seguido para proponer el nuevo proceso, así como la metodología de validación.

(Maida & Pacienzia, 2015), presentan una relación detallada de los procesos tradicionales estructurados y los procesos ágiles, resaltando los paradigmas que marcan la diferencia entre ambos tipos de procesos e identificando el que más se adapta a un proyecto determinado, permite identificar qué procesos se adaptan a proyectos específicos, sin embargo, cabe resaltar que faltó analizar los procesos en el marco del desarrollo unipersonal.

(Brito Abundis, 2013), hace un análisis de los diversos procesos de desarrollo y la manera como cada uno de ellos hace frente a las diversas amenazas a las que se enfrenta la calidad de software en el contexto actual, identificando cuales incluyen elementos de seguridad como elemento básico del proceso.

(Loboguerrero, Castañeda Bueno, & Arboleda, 2011), proponen un proceso de desarrollo de software ligero y adaptado a un contexto preciso y bien definido, alineando el PMBOK con MSF4ASD "Microsoft solutions framework for agile software development", contempla elementos de procesos de desarrollo de software ágil adaptado, que contiene factores que sirven de referente en la propuesta de proceso unipersonal.

(Agarwal & Umphress, 2008), hacen un análisis de la programación extrema (XP) para determinar la manea adecuada de aplicarla a equipos de una sola persona, denominado éste como aquel programador que trabaja por cuenta propia. Proponen entonces un proceso de desarrollo de software para una sola persona denominado Personal Extreme Programming (PXP). Este proceso constituye un acercamiento de propuesta a proceso de desarrollo unipersonal.

(Watts, 2000), PSP Personal Software Process, proporciona a los ingenieros un marco personal disciplinado para realizar trabajos de software. Contiene un conjunto de métodos, formularios y scripts que muestran como planificar, medir y administrar el trabajo de los ingenieros, junto con algunos principios generales de calidad. Constituye entonces un antecedente que fortalece la propuesta del proceso de desarrollo unipersonal, aunque se debe considerar que no abarca todo el ciclo de vida de un desarrollo.

(Pazmiño Peña, 2013), elabora una investigación que da por resultado la adaptación de un proceso ágil, para la implementación de aplicaciones, dando respuesta a la pregunta ¿es necesario aplicar un criterio de planificación para realizar una implementación exitosa usando como recurso humano solo una persona?, contiene elementos importantes que permiten dar respuesta a la pregunta planteada y que constituye un marco de referencia para el proceso que se pretende proponer.

Lo anterior evidencia el análisis detallado de los procesos de desarrollo de software dispuestos, en su mayoría para ser ejecutadas por equipos de más de 3 personas, dejando de

lado a los programadores de pequeñas aplicaciones que se desempeñan en MiPymes o ejercen como *Freelance*1. Por otra parte, las propuestas para una sola persona por lo general no abarcan todo el ciclo de desarrollo o no están alineadas con modelos de calidad. Es clara entonces la necesidad de definir un proceso que permita garantizar la calidad de los desarrollos de software efectuados de manera unipersonal.

[1] Trabajando bajo contrato para una variedad de compañías, en lugar de trabajar como empleado para una sola compañía. Se consideran independientes y tienen la libertad de escoger y elegir sus proyectos y compañías a los que les gustaría asociarse. (WebFinance Inc, 2017)

3 ÁREA PROBLEMÁTICA Y PREGUNTA DE INVESTIGACIÓN

La vertiginosa evolución de las plataformas tecnológicas ha impulsado la adopción de procesos, métodos y modelos de desarrollo de software en todo tipo de organizaciones que cada vez más se aventuran a desarrollar soluciones informáticas que permitan incrementar la competitividad.

Los desarrollos de software a lo largo de su evolución, han contado con diversos modelos de ciclo de vida, "desde la cascada hasta la aparición de las metodologías ágiles a finales de los 90", (Canós, Letelier, & Penadés, 2004), que implican de manera directa la integración de equipos de trabajo en donde cada uno de sus miembros desempeña un rol específico, delimitado y necesario para obtener productos de buena calidad atendiendo a las mejores prácticas que cada uno ellos involucran. A pesar de esto, de acuerdo con (ACIS - Asociación Colombiana de Ingenieros de Sistemas, 2016), solo "3 de cada 10 proyectos son exitosos, dentro de las causas de fallo comunes están: métodos de trabajo, manejo de alcance, soporte ejecutivo, madurez emocional, participación de los usuarios, competencias del equipo".

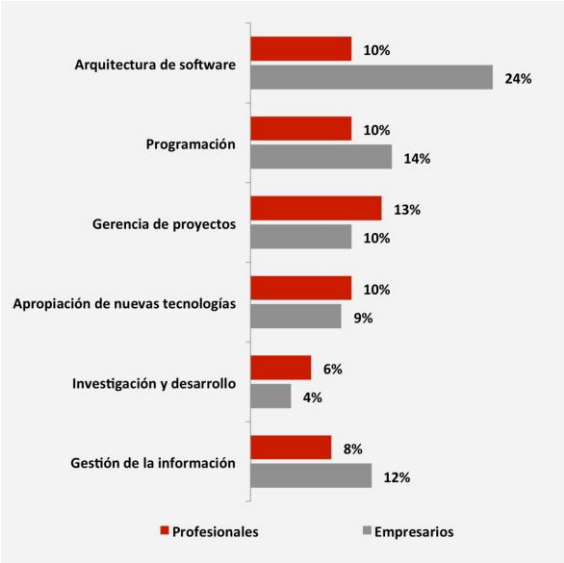
Por otra parte, de acuerdo con lo expuesto por (Mogollón Afanador, 2010), el 82% de los programadores consultados en su estudio han tenido experiencias en desarrollo unipersonal.

Gráfica 1 Experiencia en desarrollo

Fuente: (Mogollón Afanador & Estaban Villamizar, 2011)

Por su parte las estadísticas arrojadas por el Observatorio Nacional de TI, correspondientes al tercer trimestre del año 2017 , muestran una brecha de prioridades sobre conocimientos y habilidades en profesionales y empresas del sector TI en el área de programación; lo cual refuerza la necesidad de establecer un proceso de desarrollo unipersonal de calidad, teniendo en cuenta que los resultados reflejan una brecha entre lo que perciben las empresas de los profesionales de desarrollo en relación con lo que estos últimos consideran de sí mismos, tal como se observa en la gráfica 2.

Gráfica 2 Caracterización de la brecha de Talento Digital en Colombia



Fuente: (Observatorio TI, 2017)

El sector empresarial y los profesionales consideran que existe una brecha en conocimiento y habilidades que distan notoriamente de la meta del 100%.

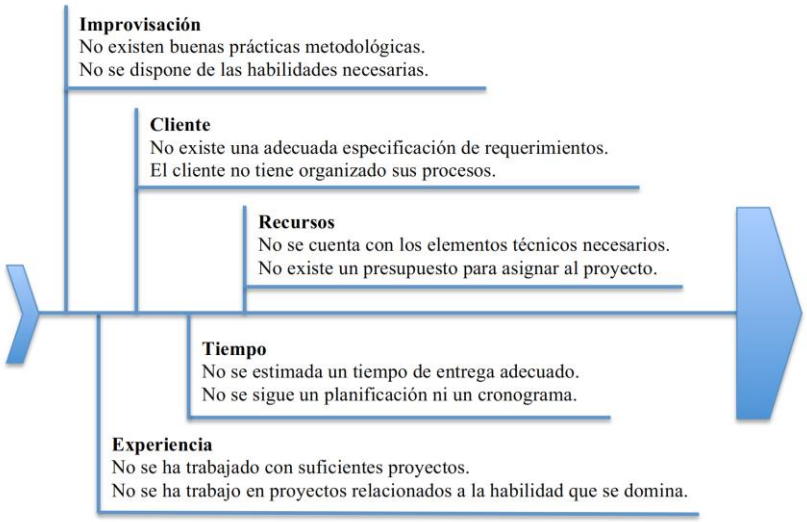
Se genera entonces la necesidad de plantear un proceso que fortalezca el desarrollo de los proyectos de software en modalidad unipersonal, en virtud de que en gran medida los desarrollos pequeños y específicos son abarcados por desarrolladores o microempresas que en la mayoría de los casos encomiendan dicha actividad a un solo individuo.

Lo anterior nos lleva a preguntarnos ¿Cómo combinar procesos ágiles y la norma ISO 29110:2014 para definir un proceso que permita desarrollar proyectos de software en modalidad unipersonal?

4 JUSTIFICACIÓN

Los procesos de desarrollo de software actualmente más utilizados se encuentran orientadas a ser implementados por equipos de 3 personas en adelante con roles definidos, por ejemplo en TSP, y en SCRUM, lo que constituye una debilidad en el proceso de desarrollo cuando no se tiene equipos de trabajo de varias personas, sino que la actividad es emprendida por una sola persona que hace todas las tareas, algunos de estos problemas descritos por (Pazmiño Peña, 2013), son: "no suelen ser tan pequeños los desarrollos como se pensó, existe improvisación, falta documentación, una metodología errónea da origen una solución errónea, cliente inconforme, esclavitud permanente con el proyecto, dentro de las más representativas" p. 17. Lo que puede evidenciarse más claramente en la gráfica 3.

Gráfica 3 Problemas de la falta de metodologías para desarrollo unipersonal



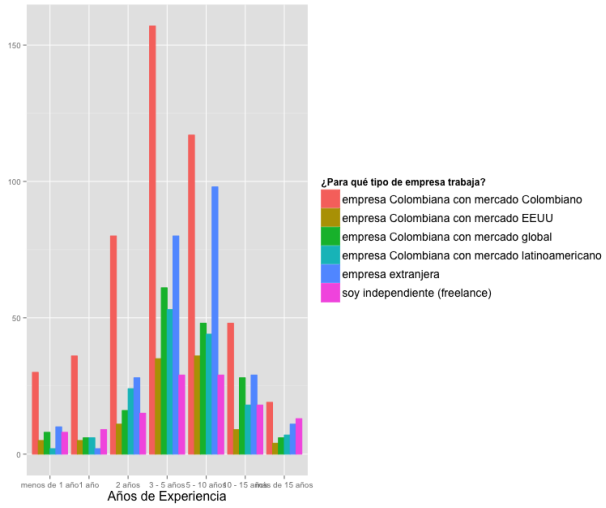
Fuente: (Pazmiño Peña, 2013, pág. 17)

Así lo expresaban (Mogollón Afanador J. O., 2010) en su tesis al decir:

en la actualidad existen innumerables metodologías de desarrollo de software, cada una con sus buenas prácticas, pero el común denominador de dichas metodologías es la necesidad de interactuar entre diferentes personas para lograr obtener un producto de software y esto implica tener más de un miembro en el equipo de desarrollo. p. 48.

De igual manera en la encuesta realizada en el año 2015 y cuyos resultados fueron publicados en el año 2016, se evidencia una gran cantidad de profesionales y no profesionales que desempeñan actividades de desarrollo de software, ver gráfica 4.

Gráfica 4 Contratación de desarrolladores de acuerdo con el tipo de empresa y experiencia

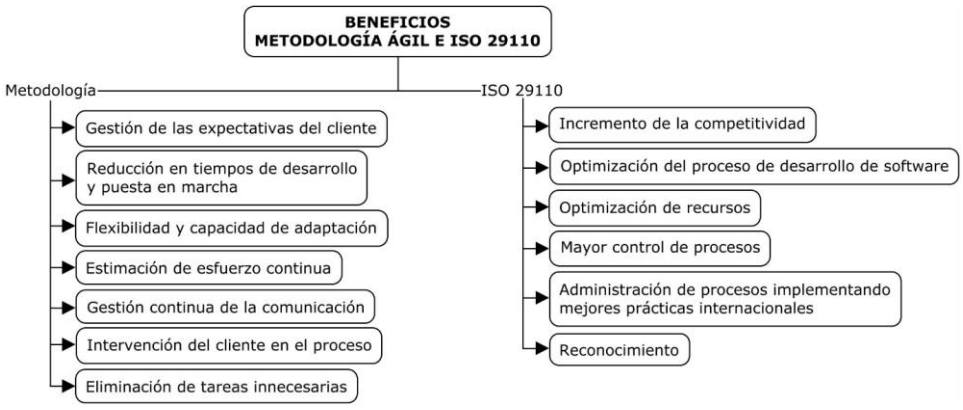


Fuente: (Colombia Dev, 2016)

En la imagen anterior se puede ver que existe una cantidad representativa de personas dedicadas a las actividades de desarrollo de software de manera *freelance*, independientemente de la cantidad de años de experiencia que se tenga en el sector.

Considerando lo anterior, puede decirse que los procesos ágiles y la norma internacional ISO 29110:2014 aportan beneficios a la actividad de desarrollo de software. Ver gráfica 5.

Gráfica 5 Beneficios procesos ágiles e ISO 29110:2014



Fuente: elaboración propia

Algunos de los beneficios de implementar un proceso ágil en el desarrollo de software son:

- Gestión de las expectativas del cliente: el cliente establece sus expectativas indicando el valor que le aporta cada requisito del proyecto y cuándo espera que esté completado.
- Reducción en tiempos de desarrollo y puesta en marcha: se entregan funcionalidades completas y funcionales
- Flexibilidad y Capacidad de adaptación: basando su ejecución en el principio de mejora.
- Estimación de esfuerzo continua: estimación del esfuerzo necesario para completar requisitos y tareas.
- Gestión continua de la comunicación: establece canales de comunicación permanentes entre los miembros del equipo de desarrollo, incluyendo el cliente.
- Intervención del cliente en el proceso: facilita la participación del cliente en el proceso de desarrollo al considerarlo un miembro más del equipo de trabajo.
- Eliminación de tareas innecesarias: solo se prioriza en las actividades que son realmente importantes, desestimando aquellas que no revisten mayor importancia, apoyando la centralización de esfuerzos.

Por su parte los beneficios de implementar los requisitos para el desarrollo de software establecidos en la norma ISO 29110:2014, de acuerdo con lo expuesto por (Laporte, 2016) son:

- Trabajo estandarizado y entregables consistentes en todos los proyectos
- Evita “reinventar la rueda” para cada nuevo proyecto
- El trabajo se realiza en forma sistemática y disciplinada
- Mejor calidad de los productos de trabajo internos y externos
- Mejor gestión de proyectos y supervisión de proyectos
- Reducción de riesgos del proyecto
- Mejor comunicación dentro del equipo

- Mejor credibilidad para ofertar proyectos
- Acceso a mercados que requieren demostración de cumplimiento a un estándar de proceso.
- Mejor reconocimiento de la calidad del trabajo y los productos
- Mejor confianza de los clientes y socios comerciales

Lo anterior refleja la necesidad de establecer un proceso de desarrollo de software unipersonal que permita a los programadores *freelance* asegurar la calidad en los productos de software y la satisfacción de sus clientes.

5 REFERENTE TEÓRICO

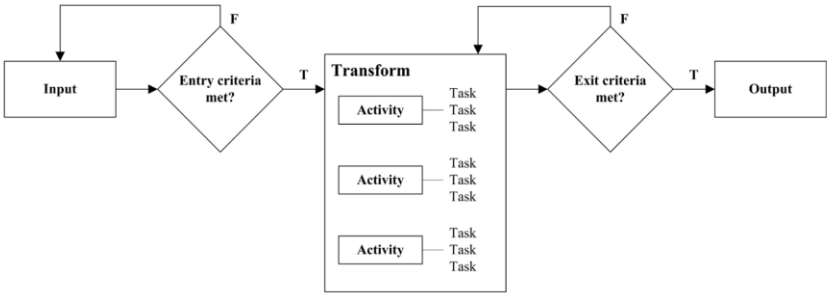
5.1 PROCESO DE DESARROLLO DE SOFTWARE

Es un conjunto de actividades y resultados asociados que producen un producto de software. Existen cuatro actividades fundamentales de procesos que son comunes para todos los procesos del software: especificación del software, diseño e implementación del software, validación del software y evolución del software, (Sommerville, 2011, pág. 28).

Es un enfoque adaptable que permite que las personas que hacen el trabajo busquen y elijan el conjunto apropiado de acciones y tareas para el trabajo. Se busca siempre entregar el software en forma oportuna y con calidad suficiente para satisfacer a quienes patrocinaron su creación y a aquellos que lo usarán, (Pressman, 2010)

Así mismo, la IEEE lo define como "un conjunto de actividades y tareas interrelacionadas que transforman entradas de trabajo en productos de salida. Como mínimo, la descripción de un proceso de software incluye entradas, transformación de actividades y resultados generados", (IEEE Computer Society, 2014, pág. 150)

Gráfica 6 Elementos de un proceso de software



Fuente: (IEEE Computer Society, 2014)

Los procesos de desarrollo de software constituyen entonces la guía metodológica que permiten obtener los resultados previstos por el trabajo de desarrollo de software definiendo de manera clara las actividades que deben realizarse de manera lógica a fin de lograr la satisfacción del cliente.

5.2 PROCESO DE DESARROLLO DE SOFTWARE[2]

"Constituyen un conjunto integrado de técnicas y métodos que permiten abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo". (Maida & Pacienzia, 2015, pág. 12)

Por lo tanto, los procesos pretenden desarrollar el software de manera sistemática en procura de lograr alta efectividad en la gestión de un proyecto de software.

"Constituye entonces un marco de trabajo que se usa para planificar y controlar el proceso de desarrollo de sistemas de información". (Maida & Pacienzia, 2015, pág. 133)

A continuación, se describen los elementos más importantes de los marcos de trabajo de desarrollo de software ágil relevantes para el desarrollo de este proyecto.

5.3 PROCESOS ÁGILES

"Conjunto de métodos de ingeniería del software, que se basan en el desarrollo iterativo e incremental, teniendo presente los cambios y respondiendo a estos mediante la colaboración de un grupo de desarrolladores auto – organizados y multi disciplinares". (Garces Guayta & Egas, 2012, pág. 7)

"Desarrollo ágil es un término derivado del Manifiesto Ágil, escrito en 2001, por un grupo que incluía a los creadores de Scrum, Extreme Programming (XP), Dynamic Development Method (DSDM) y Crystal", (Microsoft Developer Network, 2013).

(Beck, y otros, 2001) a través del Manifiesto Ágil establecieron un conjunto de cuatro valores que permiten generar equipos de alto desempeño:

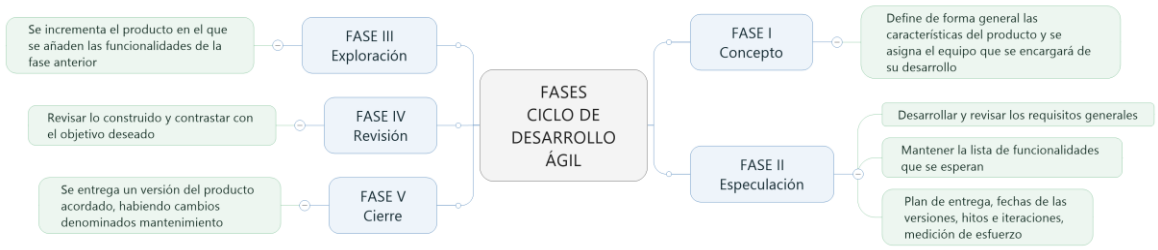
- Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación contractual.

[2] Hace algunos años se usaba el término "metodología" para lo que actualmente se conoce como "proceso". Aunque no son equivalentes, la definición de proceso está mejor estructurada y es la que usa en el cuerpo de conocimiento de la ingeniería de software (SWEBOK), por lo que es la que se trabaja en el documento.

- Respuesta ante el cambio sobre seguir un plan.

Los procesos ágiles se basan en cinco fases, ver gráfica 7, que definen el ciclo de desarrollo ágil.

Gráfica 7. Fases que definen el ciclo de desarrollo ágil



Fuente: elaboración propia

5.3.1 Extreme Programming XP

Es un proceso de desarrollo ágil que fue propuesta por Kent Beck en 1999, cuyo énfasis está puesto más en la adaptabilidad que en la previsibilidad.

"Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, e incluso deseable en el desarrollo de proyectos", (Bustamante & Rodríguez, 2014), por lo cual desde su perspectiva el cambio en los requerimientos en cualquier momento del ciclo de vida del proyecto constituye una necesidad de adaptación que conlleva a un acercamiento a la real definición de requisitos.

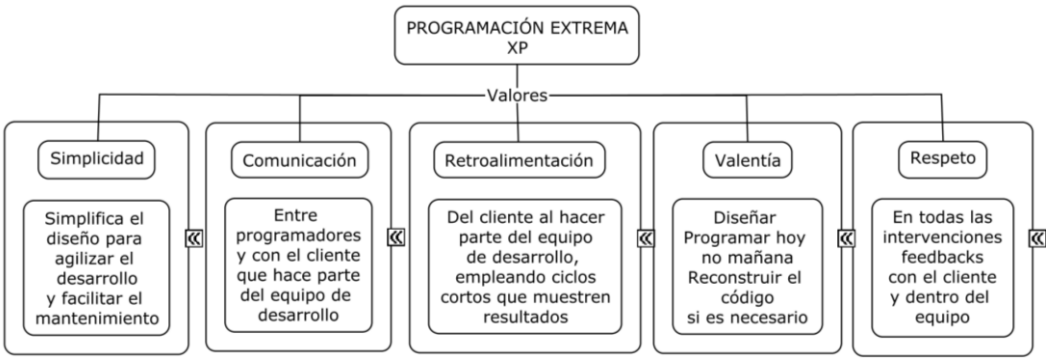
Características de XP:

- Se enfatiza en la adaptabilidad antes que en la previsibilidad
- Es dinámico durante el ciclo de vida del proyecto
- Se adapta a cambios en los requerimientos
- Los individuos son más importantes que los procesos o las herramientas
- La funcionalidad del software es más importante que su documentación detallada
- La colaboración con el cliente es más importante que la negociación de un contrato
- La respuesta a cambios es más importante que el seguimiento estricto de un plan

La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades. (Bustamante & Rodríguez, 2014, pág. 4)

La programación extrema se fundamenta los siguientes valores, ver gráfica 8

Gráfica 8. Valores programación extrema XP



Fuente: elaboración propia

La programación extrema se desarrolla por fases definidas a lo largo del ciclo de vida del proyecto de software, (Bustamante & Rodríguez, 2014), ver gráfica 9.

Gráfica 9. Fases de XP

Fuente: elaboración propia

Fase I – planificación del proyecto: consta de las siguientes actividades:

- Historias de usuario: escritos del cliente en un lenguaje no técnico sin hacer hincapié en los detalles, son usadas para estimar tiempos de desarrollo y en la fase de pruebas. El cliente y los desarrolladores se reúnen para detallar lo que tiene que hacer cada historia y el tiempo de desarrollo se estima de 1 a 3 semanas.
- Planificación de liberación: se indican las historias de usuario que se crearán para cada versión del programa y las fechas en que serán publicadas, se deben tener claros cuatro factores: los objetivos que se deben cumplir, el tiempo que tardará en desarrollarse y publicarse, el número de personas que trabajaran en el desarrollo y como se evaluará la calidad del trabajo realizado.
- Iteraciones: se divide el proyecto en iteraciones de aproximadamente 3 semanas cada una, se tiene en cuenta lo establecido en el plan de liberación, y se seleccionan aquellas historias que no pasaron las pruebas, para ser asignadas como tareas a los desarrolladores.
- Velocidad del proyecto: rapidez con la que se desarrolla el proyecto, contando el número de historias de usuario que se implementan por cada iteración.
- Programación en parejas: para mejorar la productividad y la calidad del software se involucra a dos programadores en el mismo equipo, uno con funciones de calidad de los métodos y el otro analizando su diseño.
- Reuniones diarias: se llevan a cabo por los programadores y deben participar todos los miembros del equipo de desarrollo.

Fase II – Diseño: consta de las siguientes actividades:

- Diseños simples: se procura hacer todo lo más sencillo posible, buscando un diseño entendible y fácil de implementar que no involucre sobre tiempo o esfuerzos.
- Glosario de términos: especificación adecuada de los nombres de los métodos y las clases que facilite el entendimiento por parte de todos los miembros del equipo.

- Riesgos: se analiza por parejas de programadores el impacto de los posibles problemas que surjan durante las actividades de desarrollo, de tal manera que se establezcan controles que permitan minimizar su ocurrencia o impacto.
- Funcionalidad extra: nunca incluir funcionalidad extra al desarrollo encomendado.
- Refactorizar: pequeños cambios en el código sin afectar su funcionalidad. Reutilizar códigos ya creados con el fin agilizar tiempos.

Fase III – Codificación: se desarrolla atendiendo a estándares de codificación garantizando su comprensión, consistencia y escalabilidad.

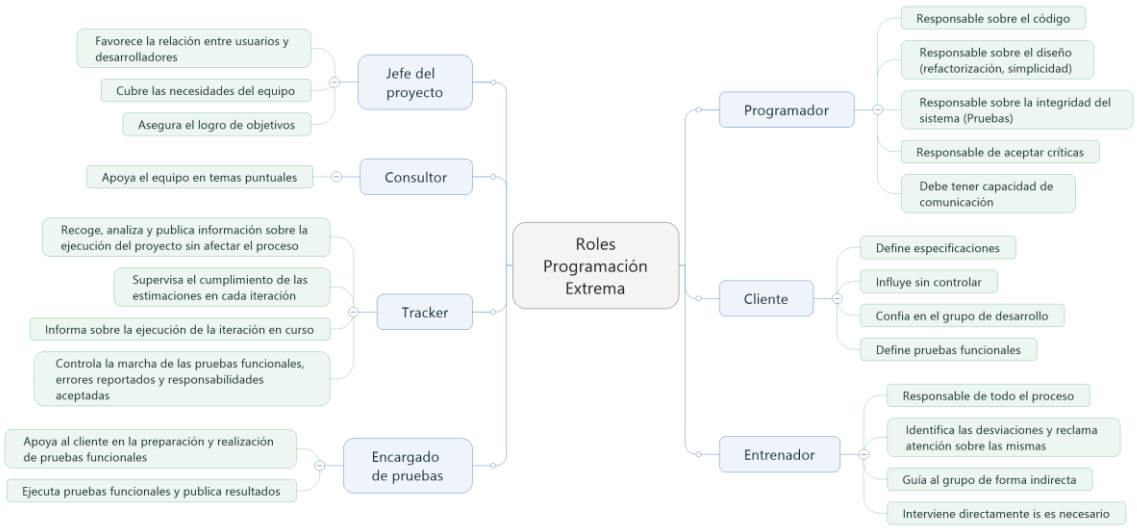
- Disponibilidad del cliente: el cliente forma parte del equipo de trabajo y "es fundamental para que el trabajo se desarrolle de manera adecuada". (Melendez Valladarez, Gaitán, & Pérez Reyes, 2016)
- Uso de estándares: "se emplean estándares de programación que facilitan la recodificación y el entendimiento del equipo de trabajo". (Melendez Valladarez, Gaitán, & Pérez Reyes, 2016)
- Programación dirigida por pruebas: se deben escribir las pruebas que el sistema debe pasar, "siendo necesario que el producto final pase las pruebas concebidas como mínimo". (Melendez Valladarez, Gaitán, & Pérez Reyes, 2016)
- Propiedad colectiva del código: todo el equipo puede contribuir con nuevas ideas en cualquier parte del proyecto.

Fase IV – Pruebas: se debe comprobar el funcionamiento de los códigos ya implementados.

- Pruebas unitarias: "se deben pasar todas las pruebas unitarias antes de ser liberadas las versiones". (Melendez Valladarez, Gaitán, & Pérez Reyes, 2016)
- Detección y corrección de errores: se corrigen los errores de manera inmediata, estableciendo mecanismos tendientes a que no vuelva a ocurrir.
- Pruebas de aceptación: se crean en base a las historias del cliente, "las que no se consideran terminadas hasta que no pasan las pruebas de aceptación". (Melendez Valladarez, Gaitán, & Pérez Reyes, 2016)

La programación extrema considera siete roles básicos: Programador, cliente, encargado de pruebas, *tracker*, entrenador, consultor y jefe de proyecto, ver gráfica 10.

Gráfica 10. Roles XP



Fuente: elaboración propia

5.3.1.1 Ventajas

De acuerdo con lo expuesto por (Borja López, 2015), el proceso XP tiene las siguientes ventajas:

- Se consiguen productos usables con mayor rapidez.
- El proceso de integración es continuo, por lo que el esfuerzo final para la integración es nulo. Se consigue integrar todo el trabajo con mucha mayor facilidad.
- Se atienden las necesidades del usuario con mayor exactitud. Esto se consigue gracias a las continuas versiones que se ofrecen al usuario.
- Se consiguen productos más fiables y robustos contra los fallos gracias al diseño de los test de forma previa a la codificación.
- Se obtiene código más simple y más fácil de entender, reduciendo el número de errores.

- Gracias a la filosofía del "pair programming" (programación en parejas), se consigue que los desarrolladores apliquen las buenas prácticas que se les ofrecen con la XP.
- Se consigue tener un equipo de desarrollo más contento y motivado. Las razones son, por un lado, el que la XP no permite excesos de trabajo (se debe trabajar 40 horas a la semana), y por otro la comunicación entre los miembros del equipo que consigue una mayor integración entre ellos

5.3.1.2 Desventajas

De acuerdo con lo expuesto por (Borja López, 2015), el proceso XP tiene las siguientes desventajas:

- Resulta muy complicado planear el proyecto y establecer el costo y la duración de este.
- No se puede aplicar a proyectos de gran escala, que requieran mucho personal, a menos que se las subdivida en proyectos más pequeños.
- Es más complicado medir los avances del proyecto, pues es muy complicado el uso de una medida estándar.

5.3.2 Scrum

En 1996, Jeff Shuterland y Ken Schwaber "presentaron las prácticas que se usaban como proceso formal para el desarrollo de software", (Trigas Gallego & Domingo Troncho, 2012).

"Scrum se basa en la teoría del control de procesos empírica o empirismo, asegurando que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce", (Schwaber & Shuterland, 2013)

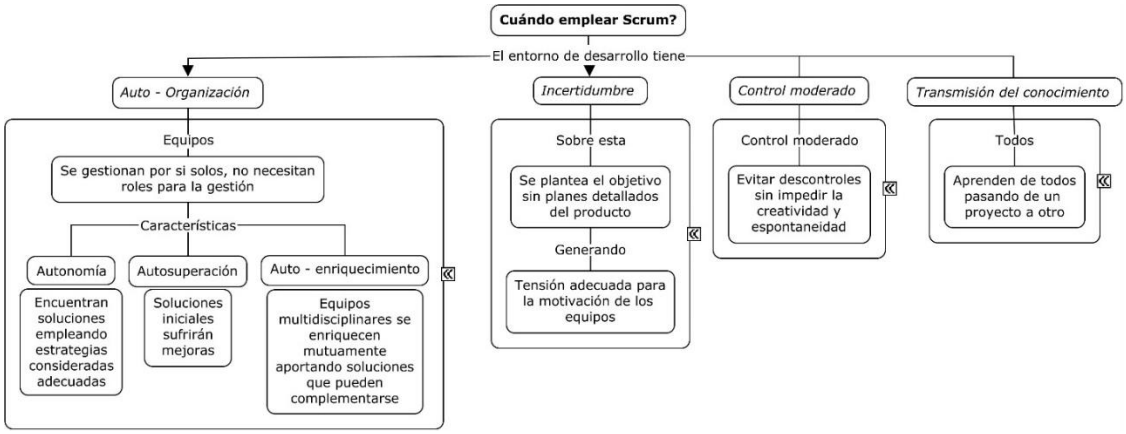
Dicha implementación del control de procesos empírica se basa en tres pilares:

- Transparencia: "los aspectos significativos del proceso deben ser visibles para aquellos que son responsables del resultado. (Schwaber & Shuterland, 2013, pág. 5)
- Inspección: "los usuarios Scrum deben inspeccionar frecuentemente los artefactos y el progreso hacia un objetivo, para detectar variaciones. Su inspección no debe ser tan frecuente para que interfiera en el trabajo". (Schwaber & Shuterland, 2013, pág. 5)

- Adaptación: “si se determina que uno o más aspectos de un proceso se desvían de límites aceptables, y que el producto resultante no será aceptable, el proceso o el material que está siendo procesado deberá ser ajustado”. (Schwaber & Shuterland, 2013, pág. 5)

De acuerdo con su concepción Scrum es recomendado en organizaciones en las que el proceso de desarrollo cumple con las características representadas en la gráfica 11.

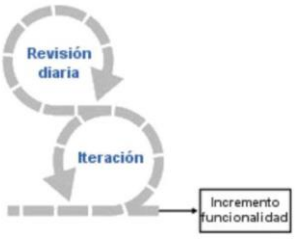
Gráfica 11. Cuando emplear Scrum



Fuente: elaboración propia

En Scrum las iteraciones de las fases de desarrollo se llevan a cabo diariamente, mediante reuniones, ver gráfica 12, dichas iteraciones se conocen como *Sprints* en este método de desarrollo de software.

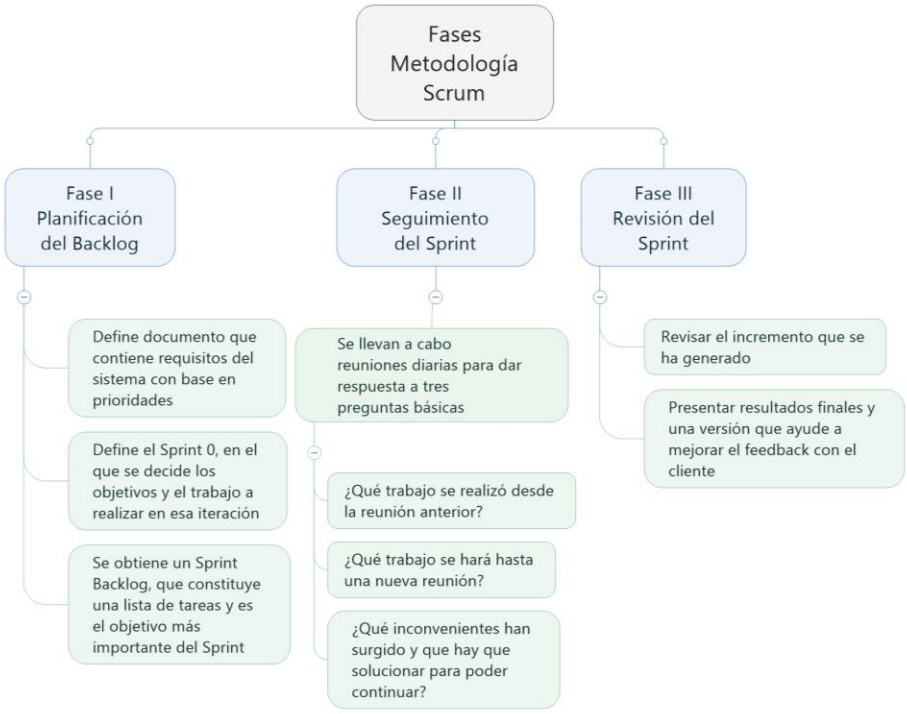
Gráfica 12. Ciclo principal de Scrum



Fuente: (Trigas Gallego & Domingo Troncho, 2012)

Scrum organiza el ciclo de desarrollo en tres fases, ver gráfica 13, las cuales constituyen artefactos metodológicos comprendidos como reuniones.

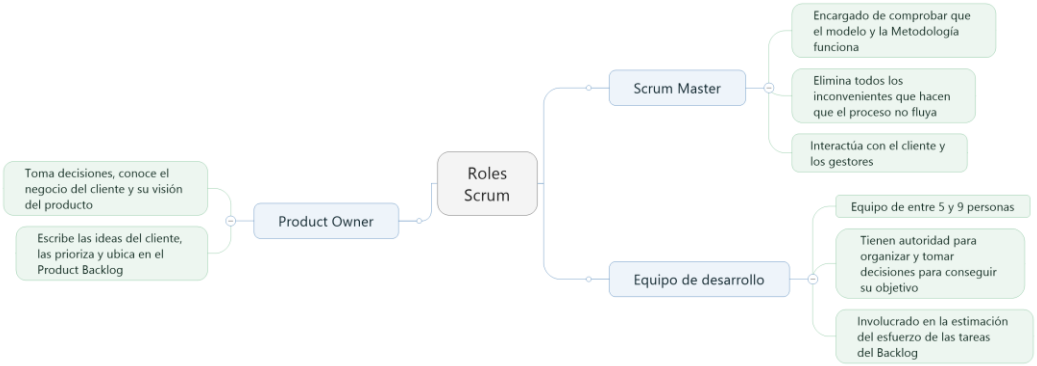
Gráfica 13. Fases de Scrum



Fuente: elaboración propia

Scrum cuenta con los roles mostrados en la gráfica 14, permitiendo la conformación de equipos de 3 a 9 personas.

Gráfica 14. Roles de Scrum



Fuente: elaboración propia

Los equipos de desarrollo de Scrum deben "ser lo suficientemente pequeños como para permanecer ágil y lo suficientemente grande como para completar una cantidad de trabajo

significativa", (Schwaber & Shuterland, 2013), indicando entonces que si el equipo de desarrollo es inferior a tres personas se reduciría la interacción y la productividad, pero por el contrario un equipo superior a nueve personas requiere demasiada coordinación y se vuelve complejo.

Dentro de los elementos que forman parte de Scrum se encuentran:

- *Product Backlog*: constituye la lista de necesidades del cliente, organizados de forma priorizada, debe incluir:
  - Objetivos del producto, usualmente expresado en historias de usuario
  - Valor que le da el cliente y el costo estimado, permitiendo una priorización de acuerdo con el retorno de inversión ROI
  - Indicar las posibles iteraciones y versiones que se han indicado al cliente
  - Posibles riesgos y las actividades o controles necesarios para mitigarlos o eliminarlos.
- *Sprint Backlog*: constituye la lista de tareas de un Sprint, asignando dichas tareas a los miembros del equipo en conjunción con el tiempo en el que se espera estén finalizadas, las cuales están organizadas conforme a las prioridades del cliente.
- *Incremento*: parte desarrollada en un Sprint, totalmente terminada y operativa, sin embargo, el cliente puede hacer cambios que considere necesarios replanteando el proyecto.

5.3.2.1 Ventajas

De acuerdo con lo expuesto por (Florez Marín & Grisales Tobón, 2014), el proceso Scrum tiene las siguientes ventajas:

- El cliente está satisfecho ya que recibe lo que necesita y esperaba
- El costo en términos de proceso y gestión es mínimo, llevando a un resultado más rápido y económico

- Ayuda a la empresa a ahorrar tiempo y dinero
- Permite realizar proyectos en los que las documentaciones de los requerimientos de negocios no están muy claras como para ser desarrolladas
- Se desarrolla rápidamente y se prueba, corrigiendo cualquier error detectado
- Hay visibilidad clara del desarrollo del proceso
- Cambios fáciles de manejar en virtud a sus Sprints cortos y la realimentación constante
- Información frecuente del proceso mediante reuniones regulares
- Las reuniones diarias hacen posible medir la productividad individual

5.3.2.2 Desventajas

De acuerdo con lo expuesto por (Florez Marín & Grisales Tobón, 2014), Scrum tiene las siguientes desventajas:

- Si no existe una fecha definitiva de finalización del proyecto es posible que se siga solicitando y añadiendo, nuevas funcionalidades
- Si una tarea no está bien definida es posible que se extienda en varios Sprints
- Si los miembros del equipo no están centrados es posible que el proyecto nunca se complete
- Está bien solo para proyectos pequeños, de rápido movimiento ya que trabaja con equipos pequeños
- Requiere miembros del equipo con experiencia
- Si se practican controles muy estrictos sobre el equipo, puede llevar a la frustración
- Impacto negativo alto cuando uno de los miembros del equipo abandona el trabajo
- El control de calidad del proyecto es difícil de implementar y cuantificar a menos que el equipo de pruebas lleve a cabo un testeo de regresión luego de cada Sprint

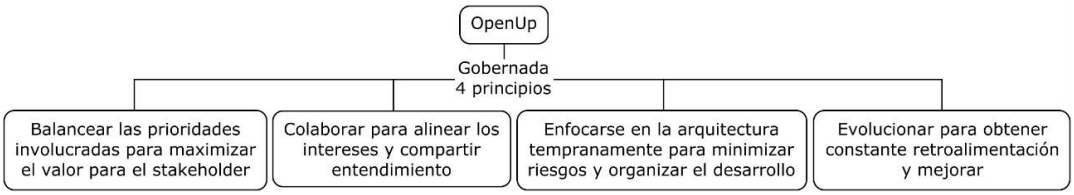
5.3.3 OpenUp

Fue propuesto por las empresas: IBM Corp., Telelogic AB., Armstrong Process Group. Inc, Number Six Software, Inc. y Xansa plc., y pasado a la fundación Eclipse en el año 2007.

Hace parte del marco de trabajo de proceso de Eclipse (EPF), manteniendo las características esenciales de RUP – Rational Unified Process. "Es un proceso ágil y liviano, que aplica enfoque iterativo e incremental dentro de un ciclo de vida estructurado y contiene un conjunto mínimo de prácticas que ayuda al equipo a ser más efectivo desarrollando software", (Gimson, 2012).

OpenUp se encuentra gobernada por cuatro principios fundamentales, ver gráfica 15

Gráfica 15. Principios de OpenUp

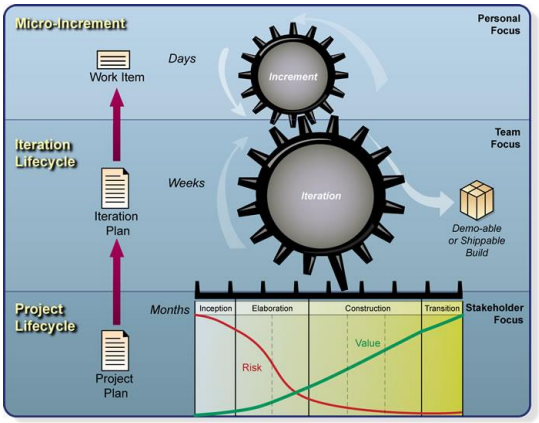


Fuente: elaboración propia

OpenUp tiene como eje central el enfoque de desarrollo colaborativo, es posible visualizarlo como un proceso completo para desarrollar software, dado su carácter incremental e iterativo que permite la liberación de versiones, sin embargo, solo incluye contenido fundamental.

La gráfica 16, muestra las capas o niveles de OpenUp

Gráfica 16 Capas de OpenUp: micro-incrementos, ciclo de vida de la iteración y del proyecto

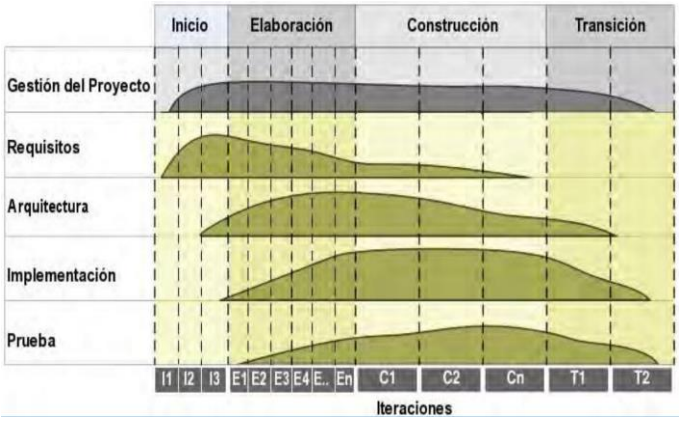


Fuente: (Eclipse, 2012)

Micro-incrementos: esfuerzo del personal aportado en un proyecto en el desarrollo de tareas granulares que contribuyen a complementar los entregables de las iteraciones, permitiendo un ritmo de desarrollo estable, así como la cuantificación del avance del proyecto, requiere personal comprometido y auto – organizado, induce a la adaptabilidad dado lo corto de los ciclos diarios de retroalimentación.

Ciclo de vida de la iteración: normalmente se dan semanalmente, y su objetivo es entregar valor incremental a las partes interesadas de acuerdo con el plan de iteración que se haya planteado, para lo cual cada equipo de desarrollo se auto – organiza con el fin de cumplir con dichos entregables, ver gráfica 17.

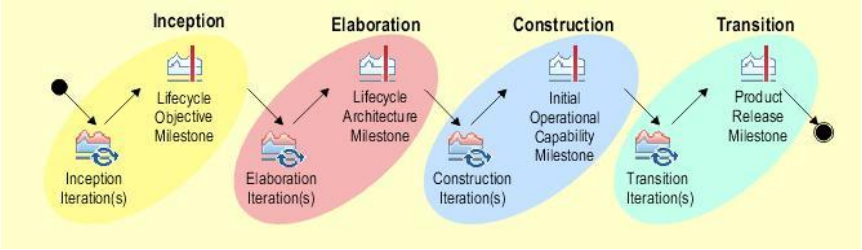
Gráfica 17. Iteraciones en el ciclo de vida de un proyecto



Fuente: (Eclipse, 2012)

Ciclo de vida del proyecto: OpenUp plantea cuatro fases en el ciclo de vida de un proyecto, ver gráfica 18.

Gráfica 18. Ciclo de vida OpenUp

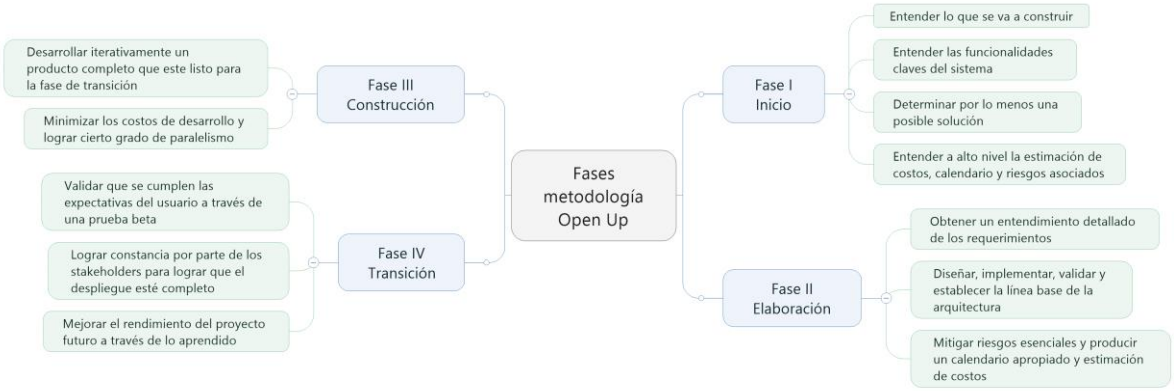


Fuente: (Eclipse, 2012)

En cada una de las fases, ver gráfica 19, pueden existir varias iteraciones, desde las primeras OpenUP se enfoca en el tratamiento de riesgos, generando reuniones que permiten definir los controles que permitan su mitigación.

- Fase de inicio: busca comprender el alcance del proyecto, objetivos, así como obtener información precisa sobre el desarrollo del sistema, permitiendo de esta manera un acuerdo entre *stakeholders* en torno a los objetivos del proyecto.
- Fase de elaboración: Cobran importancia los riesgos, teniéndolos en cuenta en toda la fase de acuerdo con su prioridad, permitiendo establecer una línea base de la arquitectura del desarrollo.
- Fase de construcción: permite completar el desarrollo del proyecto, con base en la arquitectura definida en la fase previa, permitiendo el diseño, implementación y prueba de funcionalidades.
- Fase de transición: busca garantizar que el producto se encuentra listo para ser desplegado a los usuarios, midiendo la satisfacción de las expectativas.

Gráfica 19. Fases de OpenUp



Fuente: elaboración propia

OpenUp describe mínimo seis roles que pueden acoplarse bien en el desarrollo del proceso y que requieren una interacción personal diaria, dentro de estos se encuentran: *stakeholders*, analistas, desarrolladores, arquitectos, administradores de proyecto y *testers*, ver gráfica 20, tienen autonomía para la toma de decisiones sobre las tareas a desarrollar, prioridades y cumplimiento de objetivos.

Gráfica 20. Roles principales en OpenUp

Fuente: (Eclipse, 2012)

- *Project Manager* – Administrador del proyecto: "lidera la planificación, coordina interacciones con los *stakeholders*, y mantiene al equipo enfocado en alcanzar los objetivos del proyecto", (Eclipse, 2012).
- *Analyst* – Analista: "Representa las preocupaciones del cliente y usuario final, mediante la recopilación de aportes de *stakeholders* para entender el problema a resolver", (Eclipse, 2012).
- *Arquitect* – Arquitecto: "Define la arquitectura de software, incluyendo la toma de decisiones técnicas claves que limitan el diseño y la implementación del sistema", (Eclipse, 2012).
- *Tester*: "Responsable de las actividades de pruebas, incluyendo identificación, definición, implementación y realización de las pruebas, así como el registro y análisis de resultados", (Eclipse, 2012).
- *Stakeholders*: "Representa grupos cuyas necesidades deben ser satisfechas por el proyecto. Son aquellos que se verán afectados directamente con el resultado del proyecto", (Eclipse, 2012).
- *Developer* – Desarrollador: "Responsable de desarrollar una parte del sistema, incluyendo un diseño ajustado a la arquitectura, prototipos de interfaz de usuario y

pruebas de unidad, así como de la integración de los componentes de la solución", (Eclipse, 2012).

5.3.3.1 Ventajas

De acuerdo con lo expuesto por (Gimson, 2012) y (Celis Mendoza, 2014), el proceso OpenUp tiene las siguientes ventajas:

- Se acopla a proyectos pequeños
- Se puede adaptar con otros procesos
- Permite detectar errores tempranos a través del ciclo iterativo.
- Promueve la colaboración del equipo de trabajo, alineando intereses y compartiendo conocimiento.
- Ayuda al equipo a enfocarse en la arquitectura de forma rápida, minimizándolos riesgos y organizando el desarrollo.
- Permite a los administradores del proyecto realizar un seguimiento fácil y eficaz de los avances del desarrollo.

5.3.3.2 Desventajas

De acuerdo con lo expuesto por (Gimson, 2012), OpenUp tiene las siguientes desventajas:

- No tiene mucho formalismo y se puede caer fácilmente en el desorden y perder trazabilidad del proyecto.
- Un cambio en un requerimiento importante arquitectónicamente puede retrasar todo el desarrollo del sistema.
- Se puede omitir contenido que es de interés para el proyecto.
- Se espera que cubra un amplio sistema de necesidades para los proyectos de desarrollo en un plazo muy corto.
- No es adecuado para proyectos de gran tamaño.

5.3.4 Kanban

Tiene origen en los procesos de producción de Toyota, denominados JIT (Just In Time), en los cuales se empleaban tarjetas visuales para identificar las necesidades de la cadena de producción.

“Kanban se basa en la idea de que el trabajo en curso debería limitarse, y sólo deberíamos empezar con algo nuevo cuando un bloque de trabajo anterior haya sido entregado o ha pasado a otra función posterior de la cadena”, (Maida & Pacienzia, 2015)

Para hacer el seguimiento de las actividades o tareas, Kanban emplea mecanismos de control visual que muestran el avance del flujo de valor, generando de esta manera señales que dan a conocer que existen actividades nuevas que pueden ser iniciadas.

Proporciona el valor de transparencia al mostrar claramente los problemas que pueden presentarse en el flujo del proceso de desarrollo, permitiendo la solución oportuna de los factores que bajan el rendimiento o que pueden afectar los tiempos de entrega.

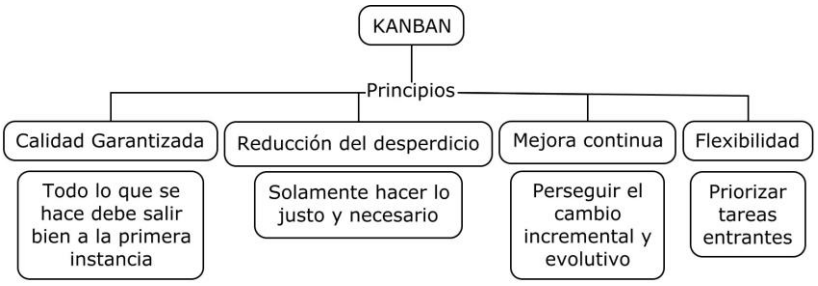
A través de su sistema de tarjetas de flujo de trabajo o tablero de trabajo, ver gráfica 21, motiva a la colaboración de todos los miembros del equipo de trabajo a la vez que la discusión de posibles mejoras, propiciando de esta manera la evolución incremental.

Gráfica 21. Tablero de Kanban

Fuente: (Bahit, 2011)

Kanban tiene los siguientes principios, ver gráfica 22:

Gráfica 22. Principio de Kanban

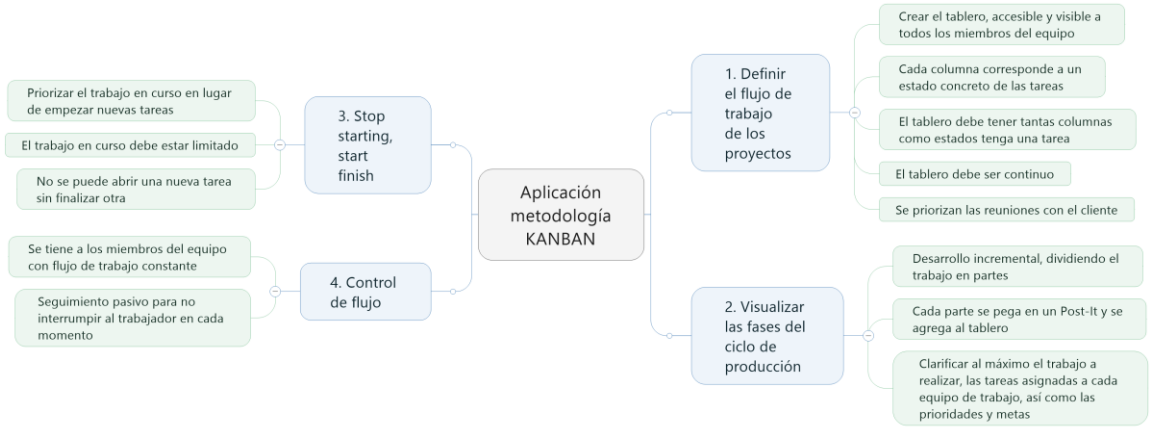


Fuente: elaboración propia

- Calidad garantizada: "todo lo que se hace debe salir bien a la primera instancia, no hay margen para el error", (Maida & Pacienzia, 2015, pág. 91). Se reconoce la calidad de los productos entregados, pues la premisa es el costo de la no calidad asociada a reprocesos.
- Reducción del desperdicio: "se basa en hacer solamente lo justo y necesario, para garantizar que se haga bien", (Maida & Pacienzia, 2015, pág. 91). Por lo tanto, las actividades que no revisten mayor importancia o que se consideran secundarias se aplazan o eliminan.
- Mejora continua: "se acuerda perseguir el cambio incremental y evolutivo", (Maida & Pacienzia, 2015, pág. 91). El equipo de desarrollo se pone de acuerdo en que el cambio gradual es la ruta que se debe seguir para aplicar cambios al sistema.
- Flexibilidad: "es necesario poder priorizar aquellas tareas entrantes según las necesidades del momento y tener la capacidad de dar respuesta a tareas imprevistas", (Maida & Pacienzia, 2015, pág. 91)

La aplicación del proceso Kanban, tiene en cuenta los aspectos representados en la gráfica 23.

Gráfica 23. Aspectos para la aplicación de Kanban



Fuente: elaboración propia

5.3.4.1 Ventajas

De acuerdo con lo expuesto por (Maida & Pacienzia, 2015), el proceso Kanban tiene las siguientes ventajas:

- Fácil de aplicar, actualizar y asumir por el equipo de desarrollo
- Gestión de tareas visual y práctica
- Aumento de la eficiencia en los procesos
- Evita retrasos y reduce los tiempos muertos
- No desaprovecha recursos, y evita sobrecarga de trabajo
- Mejor mantenimiento
- Información rápida y precisa
- Minimización de entregas con errores

5.3.4.2 Desventajas

De acuerdo con lo expuesto por (López Zamarrón, Padilla Perea, Moran, & Berrelleza, 2017), Kanban tiene las siguientes desventajas:

- Menor efectividad en situaciones de recursos compartidos: las órdenes no frecuentes vuelven ineficientes a Kanban ya que se tiene que asegurar una producción suficiente por parte de un proceso mientras que a la vez el proceso que es no frecuente es ejecutado
- Asume sistemas de producción repetitivos dada la naturaleza de su creación en el área de manufactura
- Puede arrojar productos de baja calidad que requieren reprocesos.
- Funciona a manera de semáforo para administrar el tráfico y así cumplir con las necesidades del cliente
- Cualquier evento no esperado puede afectar el funcionamiento del sistema

5.4 PROCESO DE SOFTWARE PERSONAL PSP

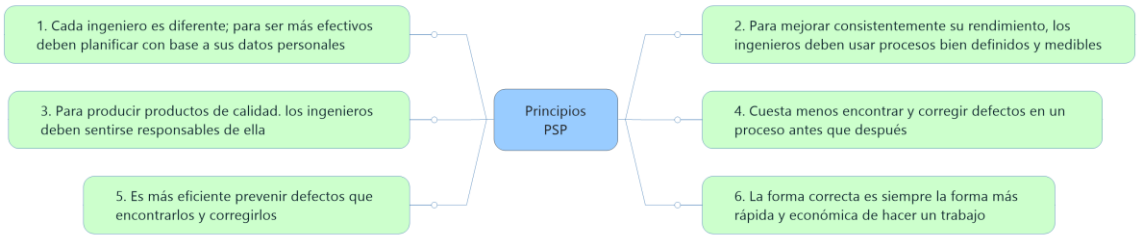
De acuerdo con (Watts, 2000), PSP "provee a los ingenieros un marco de trabajo para desarrollar las actividades de software de manera disciplinada", se encuentra conformado por un conjunto de métodos, formularios y scripts que les permiten a los profesionales planear, medir y administrar su trabajo, en búsqueda de generar productos con cero defectos.

Se tiene entonces la concepción de que el desarrollador debe seguir un plan cuidadosamente con el fin de lograr los objetivos propuestos, evitando el consumo de recursos de tiempo, humanos y económicos en la reparación de errores durante las pruebas una vez el software ha sido desarrollado.

"El principio detrás de PSP es que, para producir sistemas de software de calidad, cada ingeniero que trabaja en el sistema debe hacer un trabajo de calidad", (Watts, 2000, pág. 15).

Los principios de planificación y calidad en los cuales se basa PSP se observan en la gráfica 24.

Gráfica 24. Principios de PSP



Fuente: elaboración propia

El costo del personal representa más del 70% de los costos de un producto de software, por esta razón la productividad y los hábitos de los ingenieros de software determinan en un gran porcentaje el resultado del proceso de desarrollo de software. (Soto Duran & Reyes Gamboa, 2010).

PSP tiene los siguientes objetivos:

- Mejorar las estimaciones
- Mejorar la planeación y acompañamiento de cronogramas
- Proteger contra el exceso de compromisos
- Crear un compromiso personal con la calidad

El modelo de PSP se encuentra concebido en niveles (ver tabla 1) que se emplean de manera incremental. “Los niveles superiores adicionan características a los niveles ya implantados lo que minimiza el impacto de los cambios en los hábitos del programador”, (Soto Duran & Reyes Gamboa, 2010).

Tabla 1. Niveles de mejoramiento de PSP

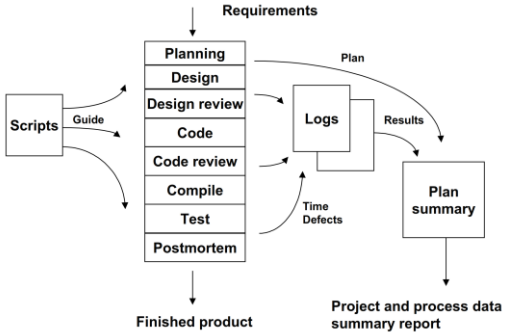
Nivel	Nombre	Actividad
PSP0	Medición personal	Registro de tiempo
		Registro de defectos
		Patrón de tipos de defectos
		Patrón de codificación
		Medida de tamaño
		Propuesta de mejoramiento de procesos
PSP1	Planeamiento personal	Estimación de tamaño
		Informe de pruebas
		Planeamiento de tareas
		Cronogramas
PSP2	Gerenciamiento de la calidad personal	Revisiones del código
		Revisiones del proyecto
		Patrones del proyecto
PSP3	Proceso personal cíclico	Desarrollo cíclico

Fuente: (Watts, 2000)

(Watts, 2000), expone las siguientes fases de PSP (ver flujo detallado de PSP gráfica 25):

- Planeación
- Diseño
- Revisión del diseño
- Codificación
- Revisión del código
- Compilación
- Pruebas
- Post mortem

Gráfica 25 Flujo de proceso PSP



Fuente: (Watts, 2000)

5.4.1 Ventajas

De acuerdo con lo expuesto por (Turriza, 2010), el proceso PSP tiene las siguientes ventajas:

- Estimación más precisa de tiempos, costos y recursos
- Cumplimiento de los compromisos
- Productividad en aumento
- Localización de los defectos desde fases iniciales
- Mejora los tiempos del ciclo de vida
- Reduce costos
- Sensación de logro
- Tomar control del propio trabajo
- Facilita los seguimientos a procesos
- Los desarrolladores comprenden su condición actual y obtienen ambiente y disciplina que propicia la mejora de su capacidad

5.4.2 Desventajas

De acuerdo con lo expuesto por (Turriza, 2010), PSP tiene las siguientes desventajas:

- Puede considerarse un proceso burocrático porque genera documentación
- Puede generar exageración en su aplicación

- Su implementación puede consumir mucho tiempo extra
- Renuencia de los desarrolladores de aplicar la documentación por sentir que se exponen al evidenciar sus tiempos de desarrollo y defectos
- Puede generar la idea de ser un proceso muy costoso en el corto y mediano plazo
- Tensión emocional de los desarrolladores por sentirse controlados

5.5 ISO 29110:2014

Contiene los perfiles del ciclo de vida para las pequeñas entidades (de máximo 25 personas) dedicadas al desarrollo de software.

Gráfica 26. Serie ISO/IEC 29110

Fuente: (ICONTEC, 2014)

"La serie ISO/IEC 29110, ha sido desarrollada para mejorar la calidad de los productos y servicios, y el desempeño de los procesos. No es su intención evitar el uso de diferentes ciclos de vida como: cascada, iterativo, incremental, evolutivo o ágil", (ICONTEC, 2014), ver gráfica 26.

Es norma está compuesta por cinco partes de acuerdo con el público objetivo, tal como se observa en la tabla 2.

Tabla 2. Público objetivo de la norma ISO/IEC 29110:201.

ISO/IEC 29110:2014	Título	Público objetivo
Parte 1	Visión general	Pequeñas entidades (PEs), productores de normas, proveedores de herramientas y proveedores de metodologías
Parte 2	Marco de trabajo y taxonomía	Productores de normas, proveedores de herramientas y metodologías
Parte 3	Guía de evaluación	Asesores y Pequeñas entidades (PEs)
Parte 4	Especificaciones de perfil	Productores de normas, proveedores de herramientas y metodologías
Parte 5	Guía de gestión e ingeniería	Pequeñas entidades (PEs)

Fuente: (ICONTEC, 2014)

Cabe anotar que para este proyecto se usó como referente el perfil básico del grupo de perfiles genéricos.

Es importante notar que la norma ISO/IEC 29110:2014 establece que varios de los roles dentro de la ejecución del proyecto pueden ser desempeñados por una sola persona o que un rol puede ser asumido por varias personas.

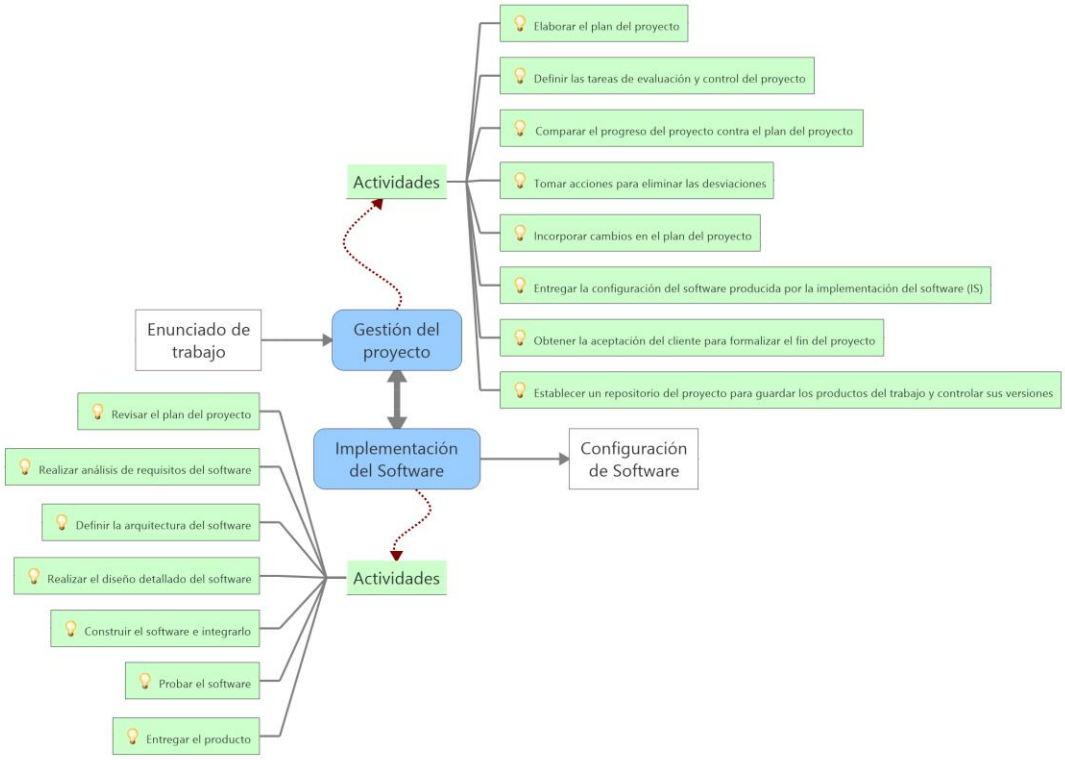
La norma "proporciona los procesos de gestión de proyecto e implementación de software, las cuales integran prácticas basadas en la selección de elementos de la ISO/IEC 12207:2008 y la ISO/IEC 15289:2006", (ICONTEC, 2014). Las pequeñas entidades pueden establecer los mecanismos necesarios para implementar cualquier proceso de desarrollo incluyendo los ágiles.

Se encuentra conformada por los siguientes procesos, ver gráfica 27:

- Proceso gestión del proyecto (GP): cuyo propósito es "establecer y llevar a cabo de manera sistemática las tareas del proyecto de implementación del software, que permiten cumplir con los objetivos del proyecto en calidad, tiempo y costo esperados", (ICONTEC, 2014), involucra las siguientes actividades:
    - Elaborar el plan del proyecto
    - Definir las tareas de evaluación y control del proyecto
    - Comparar el progreso del proyecto contra el plan del proyecto

    - Tomar acciones para eliminar las desviaciones
    - Incorporar cambios en el plan del proyecto
    - Entregar la configuración del software producida por la implementación del software (IS)
    - Obtener la aceptación del cliente para formalizar el fin del proyecto
    - Establecer un repositorio del proyecto para guardar los productos del trabajo y controlar sus versiones
- Proceso de implementación de Software (IS): cuyo propósito es "realizar sistemáticamente las actividades de análisis, diseño, construcción, integración y pruebas para productos de software nuevos o modificados de acuerdo con los requisitos especificados", (ICONTEC, 2014), involucra las siguientes actividades:
    - Revisar el plan del proyecto
    - Realizar análisis de requisitos del software
    - Definir la arquitectura del software
    - Realizar el diseño detallado del software
    - Construir el software e integrarlo
    - Probar el software
    - Entregar el producto

Gráfica 27. Procesos de la guía del perfil básico ISO/IEC 29110:2014



Fuente: adaptado de (ICONTEC, 2014)

Cada uno de estos procesos se encuentra ligado a los objetivos mostrados en la tabla 3.

Tabla 3. Relación de objetivos y procesos de la guía de perfil básico

Proceso	Objetivo	Elementos
GP – Gestión del Proyecto	GP.01: El plan del proyecto para la ejecución es desarrollado de acuerdo con el enunciado del trabajo y revisado y aceptado por el cliente. Las tareas y los recursos necesarios para completar el trabajo son dimensionados y estimados GP.02: El avance del proyecto es monitoreado contra el plan del proyecto y registrados en el registro de estado del avance. Las correcciones para resolver problemas y desviaciones respecto del plan son realizadas cuando los objetivos del proyecto no son logrados. El cierre del proyecto es ejecutado para conseguir la aceptación documentada del cliente en el documento de aceptación GP.03 Las solicitudes de cambio son atendidas mediante su recepción y análisis. Los cambios a los requisitos de software	Proceso de planeación del proyecto Proceso de medición Proceso de evaluación y control del proyecto Proceso de medición Proceso de apoyo para la aceptación del software Proceso de resolución de problemas de software Proceso de análisis de requisitos de software

	son evaluados por su impacto técnico, en costo y en el cronograma	
	GP.04 Reuniones de revisión con el equipo de trabajo y el cliente son realizadas. Los acuerdos que surgen de estas reuniones son documentados y se les hace seguimiento	Proceso de revisión del software
	GP.05 Los riesgos son identificados en el desarrollo y durante la realización del proyecto	Proceso de gestión de riesgos Proceso de revisión de software
	GP.06 Una estrategia de control de versiones de software es desarrollada. Los elementos de configuración del software son identificados, definidos e incorporados a la línea base. Las modificaciones y liberaciones de los elementos son controlados y puestos a disposición del cliente y el equipo de trabajo. El almacenamiento, la manipulación y la entrega de los elementos son controlados	Proceso de gestión de la configuración del software
	GP.07 El aseguramiento de calidad del software es realizado para proporcionar garantía de que los productos y procesos de trabajo cumplen con el plan del proyecto y especificación de requisitos	Proceso de aseguramiento de la calidad de software
IS – Proceso de Implementación del Software	IS.O1 Las tareas de las actividades son realizadas a través del cumplimiento del proyecto actual	
	IS.O2 Los requisitos del software son definidos, analizados para su correctitud y testeabilidad, aprobados por el cliente, incorporados a la línea base y comunicados	
	IS.O3 La arquitectura y diseño detallado del software son desarrollados e incorporados a la línea base. Aquí se describen los componentes del software y sus interfaces internas y externas. La consistencia y trazabilidad de los requisitos de software son establecidos	
	IS.O4 Los componentes del software definidos por el diseño son producidos. Las pruebas unitarias son definidas y ejecutadas para verificar la consistencia de los requisitos y el diseño. La trazabilidad de los requisitos y el diseño son establecidas	
	IS.O5 El software es producido ejecutando la integración de los componentes de software y es verificado utilizando los casos de prueba y procedimientos de prueba. Los resultados son registrados en el reporte de pruebas. Los defectos son corregidos y la consistencia y trazabilidad hacia el diseño de software son establecidos	

	IS.O6 La configuración del software, que cumpla con la especificación de requisitos según lo acordado con el cliente, que incluye documentación de usuario, operación y mantenimiento es integrada, incorporada a la línea base y almacenada en el repositorio del proyecto. Las necesidades de cambios en la configuración de software son detectadas y las solicitudes de cambio relacionadas son iniciadas
	IS.O7 Las tareas de verificación y validación de todos los productos de trabajo requeridos son realizados utilizando los criterios definidos para lograr la coherencia entre los productos de entrada y salida en cada actividad. Los defectos son identificados y corregidos; los registros son almacenados en los resultados de verificación / validación

Fuente: elaboración propia

Cada actividad de los procesos anteriormente mencionados se vincula con los objetivos propuestos, ver gráficas 28 y 29

Gráfica 28. Relación de actividades y objetivos del proceso Gestión del Proyecto

Fuente: elaboración propia

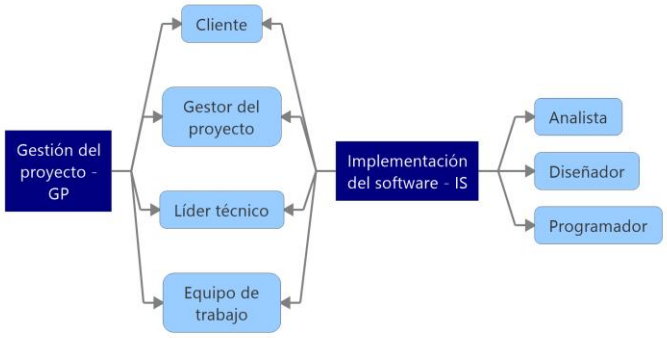
Gráfica 29. Relación de actividades y objetivos del proceso Implementación del Software

Fuente: elaboración propia

La norma ISO/IEC 29110:2014 contempla siete roles, ver gráfica 30, los cuales intervienen de acuerdo con el proceso (GP – IS):

- Analista: debe tener conocimiento en especificar y analizar requisitos, diseño de interfaces de usuario, criterios ergonómicos, técnicas de revisión, técnicas de edición y experiencia en diseño y desarrollo de software.
- Cliente: debe tener autoridad para aprobar los requisitos y sus cambios, conocimiento de los procesos del cliente y habilidad para explicar sus requisitos.
- Diseñador: Debe tener conocimiento en componentes de software, diseño de arquitectura, planificación y ejecución de pruebas de integración, técnicas de edición y experiencia en desarrollo y mantenimiento de software.
- Programador: Debe tener conocimientos en programación, integración, pruebas unitarias, técnicas de revisión, técnicas de edición, además de experiencia en desarrollo y mantenimiento de software.
- Gestor de proyecto: debe poseer capacidad de liderazgo, experiencia en toma de decisiones, planificación, gestión de personal, delegación y supervisión, conocimiento en finanzas y desarrollo de software.
- Líder técnico: debe poseer conocimiento en el proceso y en software.
- Equipo de trabajo: debe tener conocimiento y experiencia de acuerdo con sus roles dentro del proyecto, además de conocimiento en los estándares adoptados por el cliente y/o por la pequeña empresa.

Gráfica 30. Roles de ISO/IEC 29110:2014 por Procesos



Fuente: elaboración propia

5.5.1 Beneficios

Una PE puede obtener los siguientes beneficios de la implementación de la norma ISO/IEC 29110:2014, de acuerdo con lo establecido por (ICONTEC, 2014):

- Entregar al cliente un conjunto acordado de requisitos de proyecto y productos esperados.
- Lleva a cabo un proceso disciplinado de gestión que brinda visibilidad del proyecto y acciones correctivas de problemas y desviaciones del proyecto.
- Se sigue un proceso sistemático de implementación de software que satisface las necesidades del cliente y asegura la calidad de los productos.

6 OBJETIVOS

6.1 OBJETIVO GENERAL

Definir un Proceso para el Desarrollo de Proyectos de Software en Modalidad Unipersonal Combinando ISO/IEC 29110:2014 y Procesos Ágiles.

6.2 OBJETIVOS ESPECÍFICOS

- Establecer un comparativo entre procesos ágiles e ISO 29110:2014 para identificar las mejores prácticas que serán la base de la propuesta metodológica asociada al desarrollo de proyectos de software.
- Seleccionar las mejores prácticas de los modelos comparados y modelar el proceso resultante.
- Definir guías, plantillas y formatos de entregables como soporte para la correcta aplicación del proceso propuesto.
- Validar el proceso propuesto mediante un proyecto piloto.

7 METODOLOGÍA

7.1 TIPO DE ESTUDIO

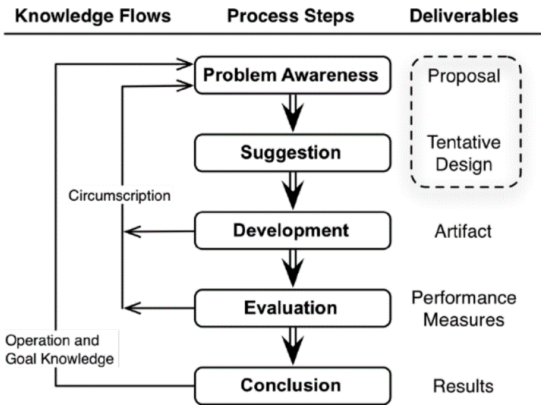
Estudio de carácter cualitativo, utilizado normalmente para solventar preguntas de investigación. Su propósito es abstraer la realidad, de la manera en que es percibida por las partes interesadas y que participan directa o indirectamente en el objeto de estudio, (Grinell, 2005).

El trabajo adopta este enfoque dado que desea generar buenas prácticas en base a los lineamientos dados por los procesos ágiles y los establecidos en la norma ISO/IEC 29110:2014, los cuales propenden garantizar oportunidad y calidad de los desarrollos de software, por tanto, los resultados se traducen en un proceso expresado en palabras, no en cifras.

Se utiliza la metodología Design Science Research (DSR) "Ciencia del diseño", de acuerdo con el modelo sugerido por (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007), ver gráfica 31.

Teniendo en cuenta que esta metodología permite desarrollar un proceso que mejora el quehacer de los desarrolladores a la vez que, indirectamente mejora la percepción de calidad y oportunidad de sus clientes.

Gráfica 31 Design Science Research



Fuente: (Helms, Remko & Giovacchini, Elia & Teigland, Robin & Kohler, & Thomas, 2010)

Como punto de partida de la investigación se ha establecido un análisis del contexto actual de los procesos de desarrollo de software unipersonal que permitan dar una mejor compresión

de la problemática, posteriormente se hará el análisis comparativo de los procesos ágiles en relación con la norma ISO 29110:2014, que permita definir estrategias para la propuesta del proceso de desarrollo de software unipersonal, el cual será validado a través de pruebas piloto de desarrollo de software.

7.1.1 Aplicación piloto del proceso

Para el presente trabajo de investigación se trabajó con una prueba piloto de desarrollo de software que permitió validar el proceso de desarrollo propuesto, empleando para ello la "escalera de pruebas", Concepto, valor y uso. A continuación, se detalla cada uno:

7.1.1.1 Prueba de concepto

Para adelantar dicha prueba se tuvieron en cuenta los siguientes elementos:

1. Definir el propósito de la prueba
2. Selección de la población de desarrolladores a encuestar
3. Selección de la forma de aplicación de la encuesta
4. Comunicación del concepto a través de EPF-Composer
5. Medición de la respuesta de los desarrolladores
6. Interpretación de los resultados

7.1.1.2 Prueba de valor

Esta prueba es aplicada con dos desarrolladores con el mismo piloto de desarrollo de software, uno de los cuales empleará su proceso habitual y el otro el proceso propuesto, determinando productividad y competitividad; se emplearán las siguientes métricas:

Tabla 4. Métricas empleadas para la prueba de valor

Métrica	Descripción
Esfuerzo	Esfuerzo requerido por el desarrollador para lograr los objetivos propuestos en el proyecto
Cubrimiento de requisitos	Permite identificar la cantidad de requisitos que se han dejado sin cubrir durante el proceso de desarrollo
Terminación de módulos en los tiempos esperados	Cumplimiento del tiempo de entrega de las versiones planificadas

Duración del proyecto	Estimación de la duración real del proyecto contrastada con la planificada.

Fuente: elaboración propia

7.1.1.3 Prueba de uso

Permite determinar si los resultados de la aplicación del proceso propuesto son los esperados. Se tendrán en cuenta las siguientes métricas del proceso de desarrollo de software.

Tabla 5. Métricas para la prueba de uso

Métrica	Descripción
Cantidad de defectos	Permite calcular la cantidad de defectos en el proceso que se han generado en una iteración
Costos de desarrollo	Determina el costo presupuestal relacionado con el esfuerzo
Fallas de requerimientos descubiertas	Identifica las fallas que se han presentado en el proceso de recolección de requerimientos
Cantidad de inconformidades relacionadas con el proceso	Retroalimentación cuantificada de las inconformidades del proceso manifestadas por el desarrollador
Puntos de historia de usuario por iteración	Permite identificar la cantidad de puntos de historia de usuario que se han desarrollado en cada iteración del proceso
Líneas de código	Cantidad aproximada de líneas de código lógicas implementadas en el desarrollo
Cantidad de documentos de soporte que se entregan	Número total de documentos que entrega el desarrollador como soporte

Fuente: elaboración propia

7.1.1.4 Técnicas E Instrumentos De Recolección De Información

Se diseñaron los siguientes instrumentos:

1. Determinación de las prácticas y principios de los procesos ágiles.
2. Comparación de las prácticas ágiles con los objetivos establecidos en la norma 29110:2014
3. Encuestas para las pruebas de concepto
4. Diseño de instrumentos para determinar las métricas del proceso

7.1.2 Procedimientos

Para alcanzar los objetivos propuestos se establecieron las siguientes fases:

Fase 1: Correspondencia de los procesos ágiles con la norma ISO 29110:2014

Consiste en el análisis de las etapas y características de cada uno de los procesos ágiles estudiados y su correspondencia con los objetivos establecidos en la norma internacional ISO/IEC 29110:2014, está fase involucra actividades tales como:

1. Revisión documental de los diferentes procesos ágiles
2. Análisis de los requisitos contenidos en la norma ISO 29110:2014
3. Aplicación de los instrumentos para analizar la correspondencia de los procesos ágiles con respecto de la norma ISO 29110:2014

Fase 2: Identificación de las mejores prácticas de los procesos analizados

1. Identificación de las mejores prácticas de los procesos de desarrollo analizados
2. Correspondencia de las mejores prácticas

Fase 3: Definición del proceso

1. Diseño de un modelo de proceso de software con base en los resultados obtenidos en las fases anteriores
2. Diseño de los procedimientos, actividades y artefactos detallados que constituirán el proceso propuesto alineado con los requisitos de la Norma ISO 29110:2014.

Fase 4: Diseño del proceso propuesto en EPF - Composer

1. Puesta a punto del proceso modelado con EPF-Composer para su distribución.

Fase 5: Validación del proceso

1. Aplicación del proceso en un piloto de desarrollo de software unipersonal
2. Aplicación de las pruebas de concepto, valor y uso
3. Análisis de resultados

7.1.3 Plan de análisis

El análisis de los resultados parciales y totales se adelantó igualmente por fases de la siguiente manera:

Fase 1: Correspondencia de los procesos ágiles con la norma ISO 29110:2014

1. Interpretación de los resultados obtenidos en la correspondencia de los procesos ágiles con relación a los requisitos establecidos en la norma ISO 29110:2014
    a. Grado de acoplamiento de los requisitos establecidos en la norma ISO 29110:2014 con relación a los procesos ágiles.

Fase 2: Identificación de las mejores prácticas de los procesos analizados

1. Identificación de mejores prácticas de cada proceso.
2. Grado de acoplamiento de los requisitos establecidos en la norma ISO 29110:2014 con relación a las mejores prácticas identificadas.

Fase 3: Definición del proceso

En esta fase se construyó una matriz que permite visualizar las mejores prácticas, en contraste con los requisitos de la Norma ISO 29110:2014.

Fase 4: Diseño del proceso propuesto en EPF - Composer

Se modeló el proceso en el software GNU/GPL EPF-Composer, incluyendo tareas, procesos, artefactos y demás herramientas que garantizan un adecuado sistema de consulta del proceso definido.

Fase 5: Validación del proceso

Se realizó la escalera de pruebas empleando pruebas de concepto, valor y uso.

1. La prueba de concepto se realizó a través de encuestas a desarrolladores lo que permitió determinar a viabilidad técnica del proceso propuesto.
2. La prueba de valor se determinó mediante la aplicación de un piloto de desarrollo de software a dos desarrolladores, de los cuales uno aplica el proceso propuesto y el otro no, lo que permitió determinar la productividad y competitividad del proceso definido, empleando para ello métricas del proceso de desarrollo de software.
3. La prueba de uso nos permitió verificar si el uso del proceso definido generaría los resultados esperados, para lo cual se aplicaron métricas del proceso de desarrollo de software.

8 RESULTADOS

8.1 ANÁLISIS DE LAS MEJORES PRÁCTICAS

Identificar y lograr apropiar las mejores prácticas es de vital importancia para obtener una ventaja competitiva; tal como decía (Szulanski, 1996) “Una práctica es un método o técnica que se emplea para realizar una parte de un proceso y describe como se realiza. Las mejores prácticas permiten incrementar la satisfacción del cliente al incorporar su uso en nuestro proceso”.

8.2 PRINCIPIOS Y PRÁCTICAS PROCESOS ÁGILES

8.2.1 Extreme programming

“La metodología XP tiene un conjunto importante de reglas y prácticas. En forma genérica se pueden agrupar en: Planificación, Diseño, Desarrollo, Pruebas”. (Wells, 1999), ver tabla 6

Tabla 6. Mejores prácticas Extreme Programming

Etapa	Descripción	Práctica	Descripción
Planificación	Diálogo continuo entre las partes interesadas.	Historias de usuarios	Escritas por el cliente, en su lenguaje, descripciones cortas de lo que el sistema debe realizar. Deben tener el detalle mínimo para estimar tiempo de desarrollo. En la implementación se conocen los detalles. Deben ser programadas entre 1 y 3 semanas; si la estimación es mayor se divide, menor se combina.
		Plan de entregas	El cronograma de entregas se desarrolla entre todas las partes interesadas y establece cuáles historias de usuario serán agrupadas para una entrega y su orden. Se desarrolla según las estimaciones de tiempo de desarrollo.
		Plan de iteraciones	Las historias de usuario seleccionadas son desarrolladas y aprobadas en un ciclo de iteración. Al comienzo de cada ciclo se hace una reunión de planificación de la iteración Cada historia de usuario es una tarea específica de programación y establece pruebas de aceptación Las pruebas se hacen al final de cada ciclo de iteración
		Reuniones diarias de seguimiento	Mantener la comunicación con el equipo y compartir problemas y soluciones Se sugiere que sean en círculo y de píe para no quitar tiempo
Diseño	Énfasis en los diseños simples y claros	Simplicidad	Un diseño simple se implementa más rápido que uno complejo, implementar el diseño simple que funcione. No adelantar funcionalidades que no corresponden a la iteración en la que se está trabajando
		Soluciones	Utilizar pequeños programas de prueba “Spike” cuando hay problemas técnicos
		Recodificación	Escribir nuevamente parte del código de un programa sin cambiar la funcionalidad, para hacerlo más simple, conciso y entendible

		Metáforas	Es algo que todos entienden sin necesidad de mayores explicaciones, se emplea para explicar el propósito del proyecto y guiar la estructura y arquitectura de este. Debe ser compartida por todas las partes interesadas
Desarrollo	Generación del código	Disponibilidad del cliente	Tener el cliente disponible durante todo el proyecto, como parte del grupo de trabajo. Los detalles de las historias de usuario son proporcionados por el cliente Puede prevenir a tiempo situaciones no deseables o funcionamientos inadecuados
		Uso de estándares	Programación basada en estándar facilitando el entendimiento y la recodificación
		Programación dirigida por pruebas	Primero se deben escribir las pruebas que el sistema debe pasar La definición de las pruebas dirige el desarrollo
		Programación en pares	Se desarrolla en pares de programadores, ambos trabajando en el mismo ordenador Minimiza los errores y mejora el diseño
		Integraciones permanentes	Todos los desarrolladores necesitan trabajar siempre con la última versión Todos los días debe haber nuevas versiones publicadas
		Propiedad del código	Todo el equipo puede contribuir con ideas nuevas que apliquen a cualquier parte del proyecto
		Ritmo sostenido	Planificar el trabajo sosteniendo un ritmo constante y razonable El trabajo extra desmotiva e impacta la calidad del producto
Pruebas	Diseño de pruebas que debe pasar el software	Unitarias	Todos los módulos deben pasar las pruebas unitarias antes de ser liberados
		Detección y corrección	Los errores se corrigen inmediatamente, se toman acciones para que no vuelvan a ocurrir
		Pruebas de aceptación	El cliente especifica los escenarios para pruebas. Se crean las pruebas en cada ciclo de iteración El cliente verifica si los resultados son los adecuados

Fuente: elaboración propia

8.2.2 Scrum

Tabla 7. Mejores prácticas y principios de Scrum

Componente	Descripción	Práctica	Descripción
Roles	Se organizan de tal manera que se maximice el valor aportado al cliente y se minimice el esfuerzo	Propietario del producto	Definir el objetivo de cada sprint con claridad, gestionar y priorizar el backlog, desglosar elementos del backlog, expresar los elementos que deben ser entendidos por el equipo de desarrollo, elaborar "roadmap" y planificar las entregas
		Equipo de desarrollo	Indicar cómo lograr los objetivos, realizar las estimaciones, actualizar el progreso diariamente, garantizar calidad del incremento
		Scrum máster	Enseña el modelo Scrum, facilita los eventos de Scrum, motiva cambios que incrementan la productividad del equipo
Artefactos	Garantizan la transparencia de información clave en la toma de decisiones	Pila del producto	Lista única de producto que recoge todo lo que necesita el producto para satisfacer las necesidades de los clientes. Los atributos de la Pila de producto son: descripción, orden, estimación y valor
		Pila del sprint	Subconjunto de elementos de la pila de productos seleccionados para abordar un plan para entregarlos como incremento en el sprint. Permite que los cambios en el progreso se entiendan en el Scrum diario
		Incremento	Resultado del sprint. Un entregable terminado, utilizable y potencialmente desplegable
Eventos	Crean regularidad y minimizan la necesidad de reuniones, además facilitan la inspección y adaptación de los aspectos del proceso, producto, progreso o relaciones	Planificación del Sprint	Define lo que se va a hacer en el sprint y cómo hacerlo, su Time-box es de 8 horas máximo para sprint de 1 mes, participan todos los miembros del equipo, buscan la definición del alcance del sprint. Define objetivo y propósito. Su entregable son los objetivos y la pila de producto para el sprint, los elementos del sprint más los pendientes constituyen el sprint backlog

			Contiene los demás eventos de scrum, garantiza la transparencia, inspección y adaptación, habilita la predictibilidad
		Sprint	Time-box máximo de 1 mes, limita el riesgo al costo de un mes Su resultado es un incremento, durante el sprint no se admiten cambios que afecten el objetivo del sprint
		Scrum diario	Evalúa diariamente el progreso y la tendencia de este, hasta finalizar el sprint. Su Time-box es de 15 minutos, se inspecciona el trabajo realizado desde el último sprint diario, se hace previsión del trabajo que puede completarse.
		Revisión del sprint	Muestra el incremento del producto y adapta la pila de producto para las próximas entregas, se realiza al finalizar el sprint, su Time-box es de 4 horas máximo, participa el equipo completo. Facilita la retroalimentación y la colaboración
		Retrospectiva	Identifica posibles mejoras al proceso y genera un plan para implementarlas, se hace después de la revisión del sprint y antes de la planificación del siguiente. Su Time-box es de 3 horas máximo
Principios	Definen un proceso simple, lógico y racional. Exigen patrones de comportamiento y control emocional	Transparencia	Aspectos claves deben ser visibles para todos. Todos deben compartir un entendimiento común
		Inspección	Usuarios deben inspeccionar frecuentemente sus artefactos y esta debe tener una frecuencia óptima que no afecte el ritmo de trabajo
		Adaptación	El proceso debe ajustarse cuando hay desviaciones y se realiza cuanto antes.
		Compromiso	Cada individuo debe estar comprometido con las metas del equipo
		Coraje	Hacer bien su trabajo, incluso en circunstancias difíciles

	Foco	Centrarse en el trabajo del sprint y en las metas del equipo
	Apertura	Los individuos deben estar abiertos al trabajo y retos que se presenten
	Respeto	Respetarse entre los miembros del equipo
	Tamaño equipo desarrollo	Mínimo 3 y máximo 9 especialistas
	Todo	El equipo de desarrollo es el único responsable de los incrementos
	Autogestión	El equipo de desarrollo decide cómo hacer su trabajo
	Autosuficiencia	Multidisciplinariedad del equipo de desarrollo
	Títulos	Todos los miembros son iguales
	Sub equipos	No existen sub-equipos en el equipo de desarrollo
	Simplicidad	Supresión de artefactos innecesarios en la gestión del sprint

Fuente: elaboración propia

8.2.3 OpenUp

Tabla 8. Mejores prácticas OpenUP

Grupo	Práctica	Descripción	Actividades
Administración	Desarrollo iterativo	Dividir el proyecto en una serie de iteraciones en recuadros de tiempo permite entregar capacidades que pueden ser evaluadas por los interesados al final de cada iteración	Planificación de iteración
			Trabajo de arquitectura de iteración
			Desarrollo continuo de micro-incrementos
			Creación de compilaciones semanales estables
			Corrección de errores
			Revisión de iteración y retrospectiva

	ciclo de vida orientado por riesgo-valor	La organización de las iteraciones en fases, con un hito definido al final de cada una, permite a los gerentes evaluar si se han cumplido los objetivos del proyecto y si este debe avanzar o no. Abordar los objetivos y riesgos en cada fase permite encontrar el equilibrio adecuado entre la reducción del riesgo y la creación inmediata de valor.	Organizar el proyecto en fases
			Proporcionar a cada fase un hito para toma de decisiones empresariales y de gestión
			Dividir las fases en iteraciones
	Plan de entregas	La planificación de versiones mejora la precisión de la planificación del proyecto. El objetivo es equilibrar la planificación de alto y bajo nivel de manera que la de bajo nivel ocurra justo a tiempo para respaldar la entrega exitosa	Comprender las características de alto nivel que se espera que el sistema posea
			Describir los planes de iteración en términos de historias de usuario, casos de uso o escenarios
			Actualizar el plan tantas veces como sea necesario para reflejar prioridades y necesidades del negocio.
	Equipo	Describe estrategias para aumentar la productividad general mediante la racionalización de la estructura de la organización del equipo y su colaboración interna	Comprensión de la filosofía de auto organización de tareas, ritmo sostenible y reuniones diarias.
			Cambios en la forma en que los proyectos son administrados y gobernados
			Discutir con el equipo el concepto de ritmo sostenible
			Identificar aspectos viables de la práctica
			Priorizar los esfuerzos de adopción
	Gestión de cambios	Debe haber algún proceso para rastrear solicitudes de cualquier miembro del equipo con relación a defectos o requisitos.	Registrar y revisar las solicitudes de cambio
			Asignar la solicitud de cambio
			Implementar la solicitud de cambio
			Verificar la solicitud de cambio

			Cerrar la solicitud de cambio
	Personalización del proceso	Guía al equipo del proyecto en la adaptación y despliegue	Tomar decisiones sobre el uso de los productos de trabajo
			Instalar, configurar y emplear las herramientas para apoyar el proyecto.
Técnicas	Pruebas continuas	Adopta pruebas continuas durante una iteración, evitando que estas se acumulen al final de la misma	Crear casos de prueba
			Implementar las pruebas
			Ejecutar las pruebas
	Integración continua	Identifica y corrige problemas de integración cuando son más fáciles de corregir, reduciendo el esfuerzo general mediante una integración frecuente	Haga los cambios disponibles con frecuencia
			Probar los cambios
			Corregir los problemas inmediatamente a su identificación
			Automatice el proceso de compilación
	Arquitectura evolutiva	Construir incrementalmente y mejorar la arquitectura del software al tiempo que descubre y aborda problemas arquitectónicos durante el desarrollo del software	Definir la arquitectura del software
			Definir qué información se debe capturar
			Decidir como equipo cómo y cuándo realizar las tareas arquitectónicas
	Diseño evolutivo	Reduce el tiempo de lanzamiento al mercado mediante la formulación incremental del diseño al implementar el software	Comprender lo nuevos detalles de los requisitos
			Identificar elementos del diseño
			Determinar cómo los elementos colaboran para realizar el escenario
			Refinar las decisiones de diseño
			Comunicar el diseño y comprender la arquitectura

			Evaluar el diseño
	Visión compartida	Permite monitorear y planificar el alcance del esfuerzo de desarrollo, proporciona la información necesaria para el análisis de costo – beneficio y la priorización del trabajo	Comprender y acordar los qué y por qué fundamentales para el producto en desarrollo
			Delinear los objetivos de alto nivel y las restricciones
			Definir el alcance y limitaciones del proyecto que son base para la priorización del trabajo y estimaciones iniciales del esfuerzo
	Desarrollo dirigido por pruebas	Mejora la productividad al encontrar y corregir errores en el momento en que se introducen y aumenta la calidad del software	Escribir las pruebas
			Ejecutar la prueba para verificar fallas
			Escribir código de implementación para ejecutar la prueba eficientemente
	Desarrollo guiado por casos de uso	Proporciona escenarios de desarrollo que expresan claramente la interacción entre los usuarios y el sistema de desarrollo	Identificar y describir los casos de uso
			Priorizar los casos de uso
Despliegue	Documentación y capacitación	Reforzar el producto antes de su publicación documentando los aspectos claves para su uso por parte de públicos múltiples y proporcionando capacitación a los usuarios finales	Generar documentación de producto, de usuario y de ayuda
			Genere los espacios de capacitación de usuarios
	Entrega de producción	Garantizar que la aplicación desarrollada esté preparada adecuadamente para entregar en producción sin consecuencias imprevistas	Planifique las tareas para la entrega a producción
			Revisar las tareas en la actividad de implementación

Fuente: elaboración propia

8.2.4 Kanban

Tabla 9. Mejores prácticas de Kanban

Prácticas	Descripción
Visualizar el flujo de trabajo	Se requiere comprender para entregar el producto solicitado. Se genera un tablero de tarjetas y columnas, donde la columna es un paso del flujo de trabajo y la tarjeta un elemento de trabajo dentro del flujo, hay 3 tipos de tarjetas, Pedido, En progreso y Hecho
	Basa el desarrollo del proyecto en el desarrollo incremental
	Pueden existir tantas fases en el tablero como sean necesarias
Eliminar las interrupciones	Establecer los límites de trabajo en progreso. Se establecen el número de elementos por Fase asegurando que se arrastran tarjetas a fases posteriores solo cuando hay capacidad disponible. Permite identificar tareas problemáticas en el flujo con el fin de tomar acciones pertinentes a su solución
Gestionar el flujo	Se debe tener flujo continuo e ininterrumpido, minimizando el riesgo y evitando costes de retraso, haciendo el trabajo previsible.
	Medir tiempo en completar una tarea, el Lead Time o cycle time cuenta desde que se hace una petición hasta que se hace una entrega.
Políticas explícitas	El proceso debe definirse, publicarse y promoverse. Garantiza el trabajo en equipo y la toma de decisiones positivas para el proyecto
Circuitos de retroalimentación	Se hacen reuniones diarias de pies que permitan sincronizar el equipo. Se llevan a cabo frente al tablero Kanban y cada miembro comparte lo que hizo el día anterior y lo que hará el día de hoy. Tiempo máximo de la reunión 15 minutos
	Reuniones para la revisión de entrega de servicios, operaciones y riesgos. Su frecuencia debe ser concretada dependiendo de los factores particulares del proyecto, se hacen regularmente, a una hora fija y nunca innecesariamente largos. Duración máxima 1 hora
Mejorar colaborando	Entendimiento compartido sobre el trabajo, el flujo, el proceso y el riesgo lo que permite sugerir acciones de mejora en consenso.

Fuente: elaboración propia

8.2.5 PSP

El Proceso de Software Personal – PSP - ha sido tenido en cuenta en virtud de la importancia que reviste como referente en procesos de desarrollo personales, al direccionar las actividades del ingeniero a la mejora continua de sus productos mediante la medición

y mejora de la productividad personal. Además, su aplicación es independiente del tipo de proceso organizacional empleado (tradicional o ágil).

Tabla 10. Mejores prácticas y principios PSP

Principios/Prácticas	Descripción
Principios de planificación y calidad	Cada ingeniero es diferente; para ser más efectivos deben planificar con base a sus datos personales
	Para mejorar consistentemente su rendimiento, los ingenieros deben usar procesos bien definidos y medibles
	Para producir productos de calidad, los ingenieros deben sentirse responsables por ella
	Cuesta menos encontrar y corregir defectos en un proceso antes que después
	Es más eficiente prevenir defectos que encontrarlos y corregirlos
	La forma correcta es siempre la forma más rápida y económica de hacer un trabajo
Diseño conceptual	Se genera un diseño aproximado de cómo se vería el producto con lo que hasta el momento se conoce
Estimar el tamaño del producto y los recursos	A partir del diseño conceptual, estimar los tamaños de los productos que se desarrollarán, luego el tiempo requerido
Realizar cronograma	Planificar el tiempo que será asignado a la realización de cada fase
Desarrollar el producto	Actividades de programación y pruebas de desarrollador
Uso del estándar personal para conteo de LOC	Contiene los siguientes elementos básicos: Base (programa que se toma como base, si no existe es cero), Adicionado (Código añadido al base), modificado (parte del código base que ha cambiado); Eliminado (Código eliminado); Reutilizado (Código perteneciente a otro programa o librería);
Definición de un estándar de tipo de defectos	Permite la eliminación temprana y prevención de defectos
Plantillas de diseño	El diseño se considera completo cuando define las cuatro dimensiones en la estructura de especificación del objeto.

Revisiones de código y revisiones de diseño	Revisar el código y el diseño con las correspondientes listas de chequeo detectando defectos inyectados, defectos removidos, defectos en la fase de entrada
Reporte de pruebas	Se utiliza un formato para registrar los resultados de las pruebas implementadas
Analizar el proceso	Realizar análisis postmorten del trabajo, incluye revisión de las medidas registradas (tiempo, tamaño y calidad). Elaborar propuestas de mejora y ajustes al proceso (PIP – Process Improvement Proposal)
Medidas de tiempo	Se debe registrar la hora de inicio, parada e interrupciones de cada fase
Medidas de tamaño	Estimar y registrar los tamaños de los productos
Medidas de calidad	Se tiene especial cuidado con los defectos, registrando el tipo, la fase en que se encontró y el tiempo de solución

Fuente: elaboración propia

8.3 Comparación de mejores prácticas procesos ágiles y su paralelo con ISO 29110:2014

La comparación realizada consistió en hacer un paralelo entre las mejores prácticas identificadas y su adopción por parte de los procesos ágiles analizados, se incluye en la tabla de comparación la columna A, que indica si el proceso emplea o no la mejor práctica identificada (marcando con una "X" si emplea la práctica), de igual manera se incluye la columna Obs, la cual establece los nombres alternativos para mejores prácticas; en la misma tabla también se identificó cuáles de las mejores prácticas se relacionan con los objetivos contenidos en la norma ISO 29110:2014, colocando una "X" si cumple con algún objetivo de la norma, y en "Obs" el identificador de objetivo correspondiente. La información resultante será utilizada como punto de partida para la incorporación de mejores prácticas en el proceso de desarrollo de software unipersonal propuesto.

Tabla 11. Comparación de mejores prácticas procesos ágiles XP, Scrum, OpenUp, Kanban, PSP y su paralelo ISO 29110:2014

N	Proceso -Norma	XP		Scrum		OpenUp		Kanban		PSP		ISO 29110:2014	
	Práctica	A	Obs	A	Obs.	A	Obs.	A	Obs.	A	Obs	A	Obs
1	Historias de usuarios	X		X	Pila del producto	X	Plan de entregas					X	IS.2
2	Plan de entregas	X		X	Pila del producto Incremento	X	Arquitectura evolutiva Diseño evolutivo	X	Visualizar el flujo de trabajo	X	PSP1	X	GP.1 GP.3 GP.4
3	Plan de iteraciones	X		X	Pila del sprint Planificación del Sprint	X	Desarrollo iterativo Ciclo de vida orientado por riesgo-valor Arquitectura evolutiva Diseño evolutivo			X	PSP3	X	GP.1 IS.4
4	Reuniones diarias de seguimiento	X		X	Scrum diario	X	Desarrollo iterativo	X	Circuitos de retroalimentación			X	GP.2 GP.3
5	Soluciones “Spike”	X											
6	Recodificación	X				X	Desarrollo dirigido por pruebas			X	PSP2	X	IS.4
7	Metáforas	X											
8	Disponibilidad del cliente	X		X		X	Visión compartida	X	Mejorar colaborando			X	GP.2 GP.4

						Desarrollo guiado por casos de uso						
9	Uso de estándares	X						X	PSP0	X	IS.3	
10	Programación dirigida por pruebas	X			X	Desarrollo dirigido por pruebas Pruebas continuas				X	IS.3 IS.4	
11	Programación en pares	X										
12	Integraciones permanentes	X	X		X	Integración continua	X	Circuitos de retroalimentación Gestionar el flujo	X	PSP2	X	IS.3
13	Propiedad del código	X										
14	Ritmo sostenido	X	X				X	Eliminar las interrupciones	X	PSP0 PSP1	X	GP.1 GP.3
15	Pruebas unitarias	X	X	Revisión del sprint Retrospectiva	X	Pruebas continuas	X	Circuitos de retroalimentación	X	PSP0 PSP1 PSP2	X	IS.4 IS.5
16	Pruebas de detección y corrección	X	X	Revisión del sprint Retrospectiva Adaptación	X	Pruebas continuas Entrega de producción	X	Circuitos de retroalimentación	X	PSP0 PSP1 PSP2	X	GP.4 IS.4 IS.5
17	Pruebas de aceptación	X	X	Revisión del sprint Retrospectiva	X	Verificación de calidad	X	Mejorar colaborando	X	PSP2	X	GP.4 IS.6

18	Propietario del producto		X								
19	Equipo de desarrollo		X	X	Equipo						
20	Scrum Máster		X			X	Políticas explícitas				
21	Sprint		X	X	Desarrollo iterativo			X	PSP3	X	IS.4
22	Transparencia		X	X	Modelación visual	X	Visualizar el flujo de trabajo				
23	Inspección		X	X	Desarrollo iterativo Pruebas continuas Entrega de producción	X	Circuitos de retroalimentación	X	PSP2	X	GP.3
24	Compromiso	X	X	X	Equipo	X	Políticas explícitas	X	PSP0		
25	Foco		X							X	IS.1
26	Tamaño equipo desarrollo		X	X	Equipo					X	GP.1
27	Autogestión		X	X	Equipo						
28	Autosuficiencia		X	X	Equipo						
29	Sub equipos		X								
30	Casos de Uso			X	Administración de requerimientos					X	IS.2
31	Control de Cambios	X	X	X	Gestión de cambios	X	Circuitos de retroalimentación	X	PSP0	X	GP.2 GP.3

32	Personalización del proceso	X		X	PSP1	X	IS.3
33	Documentación y capacitación	X		X	PSP0	X	IS.6

Fuente: elaboración propia

La cantidad total de mejores prácticas identificadas en los procesos ágiles estudiadas es de 33, que corresponden al 100% de la base total para el análisis de adopción de estas, ver tabla 12, su estudio constituye la base para el proceso de desarrollo unipersonal propuesto

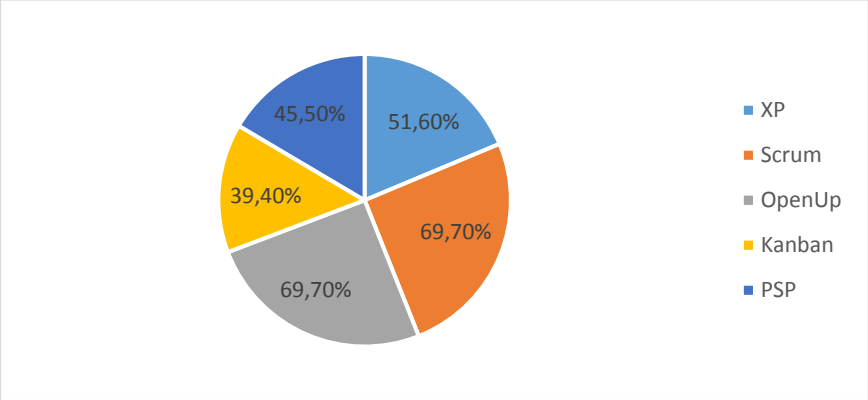
Tabla 12. Cantidad de mejores prácticas adoptadas por los procesos XP, Scrum, OpenUp, Kanban y PSP con relación al total.

Procesos	Mejores prácticas adoptadas	% Mejores prácticas identificadas adoptadas
XP	19	57,6%
Scrum	23	69,7%
OpenUp	23	69,7%
Kanban	13	39,4%
PSP	15	45,5%

Fuente: elaboración propia

Los resultados anteriores reflejan que Scrum y OpenUp son los procesos que más adoptan mejores prácticas con un 69,7%; entre tanto, la Kanban es la que menos adopta obteniendo un 39,4%, ver gráfica 32.

Gráfica 32. Adopción de mejores prácticas procesos ágiles XP, Scrum, OpenUp, Kanban y PSP



Fuente: Propia

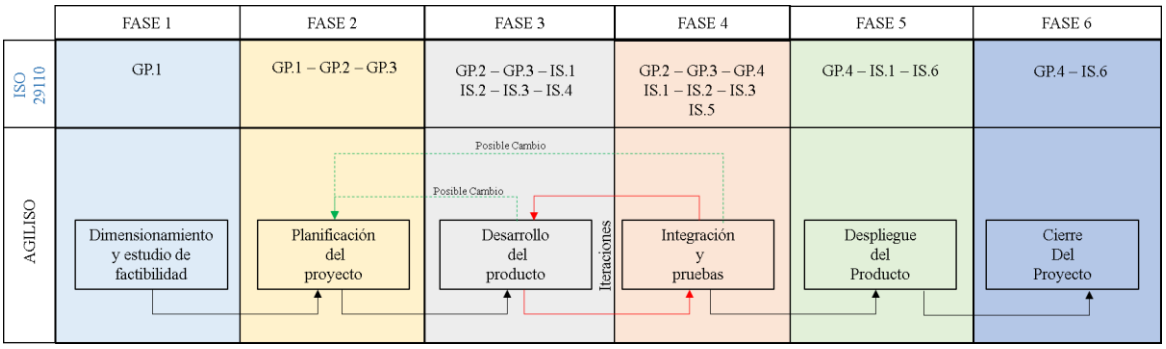
De otro lado la Norma Internacional ISO 29110:2014, se acopló en un total de 21 mejores prácticas identificadas, representando un 63,6% del total de mejores prácticas y el 100% de los procesos y objetivos definidos en la norma.

9 CONCRECIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE UNIPERSONAL

9.1 ESQUEMA GENERAL DEL PROCESO DE DESARROLLO UNIPERSONAL PROPUESTO

El proceso propuesto AGILISO se definió agrupando objetivos de los procesos de gestión y de implementación de software definidos en ISO 29110, y considerando las prácticas de los métodos analizados, que cumplieran con la norma.

Gráfica 33. Proceso de desarrollo de software unipersonal propuesto



Fuente: elaboración propia

El proceso de desarrollo unipersonal propuesto consta de seis fases, las cuales a su vez se acoplan con los objetivos de la norma ISO 29110:2014 (ver gráfica 33).

Fase 1 Dimensionamiento y estudio de factibilidad: constituye el primer acercamiento con el cliente, procura documentar y permitir el análisis de las condiciones del servicio solicitado tales como:

- Tiempo de entrega: tiempo requerido por el cliente para realizar la entrega del producto terminado, permite analizar las condiciones de disponibilidad del desarrollador con relación a otro tipo de ocupaciones que pueda estar adelantando en paralelo, tendiente a cumplir con el principio de Capacidad.
- Alcance: definición precisa con el cliente del alcance de los trabajos requeridos por parte del desarrollador, procurando dejar constancia para la posteridad de los servicios contratados, evitando desacuerdos en relación con el mismo, incluye

funcionalidades del producto que serán desarrolladas dentro del marco de la relación contractual.

- Estimación del esfuerzo: proyección del tiempo en horas/desarrollador que son requeridas para conseguir exitosamente el alcance acordado con el cliente.
- Estimación del costo: valor económico del trabajo a desarrollar teniendo en cuenta los factores de tiempo de entrega, alcance y esfuerzo estimado.

Una vez se han acordado los puntos anteriormente mencionados se constituye el acta de inicio del proyecto y se formaliza el contrato, finalizando esta fase del proceso

Fase 2: Planificación del proyecto: el desarrollador deberá iniciar el proceso de documentación de los requisitos, empleando para ello, Historias de usuario, las cuales deberán ser documentadas, independientemente de que el desarrollador decida complementar la información de requisitos empleando otra técnica de análisis de requerimientos que facilite esta etapa, de acuerdo con su experiencia y el tipo de producto a desarrollar.

El desarrollador procederá a generar un plan de entregas que será acordado con el cliente, dicho plan contendrá los alcances de las entregas, las fechas y las limitaciones encontradas durante su proceso de desarrollo.

En consecuencia, de lo anterior, el desarrollador deberá generar una planificación para las iteraciones del proceso, incluyendo en el propósito, alcance, definiciones, referencias, descripción, recursos (humanos, financieros).

Fase 3 Desarrollo del producto: el desarrollador da cumplimiento al plan de iteraciones, generado en la Fase anterior, para lo cual se deben tener en cuenta antecedentes técnicos, normativos, sistemas de información relacionados, interesados en la iteración, requerimientos funcionales, no funcionales, de interoperabilidad, de infraestructura, de seguridad, arquitectura general de la iteración, contrataciones requeridas, restricciones, oportunidades, riesgos.

Así mismo, una vez finalizada la etapa de codificación, el desarrollador deberá implementar pruebas unitarias, las desviaciones encontradas deberán ser recodificadas de manera

inmediata, permitiendo aplicarlas nuevamente, procurando que la iteración pase adecuadamente las pruebas a las cuales es sometida, cumpliendo con principio de calidad.

Es posible que resultados inesperados durante esta fase conlleven a modificaciones en la Fase 2. Planificación del proyecto en términos de extensión o disminución de tiempos de entrega afectando los cronogramas establecidos, o cambiando el alcance en término de funciones desarrolladas.

Se debe hacer monitoreo de cumplimiento del plan de entregas y el plan de cada iteración con relación a los objetivos logrados estableciendo un porcentaje de avance en el cumplimiento de estos (se sugiere emplear para ello checklist).

Fase 4 Integración y pruebas: El desarrollador debe integrar el producto, a fin de someterlo a pruebas finales que permitan evaluar su comportamiento una vez integrados todos los componentes, cuando el resultado sea exitoso, debe someter el producto a las pruebas de aceptación en conjunción con el cliente.

Es posible que resultados inesperados durante esta fase conlleven a modificaciones en la Fase 2. Planificación del proyecto en términos de cambios en tiempos de entrega o en el alcance, afectando los cronogramas establecidos.

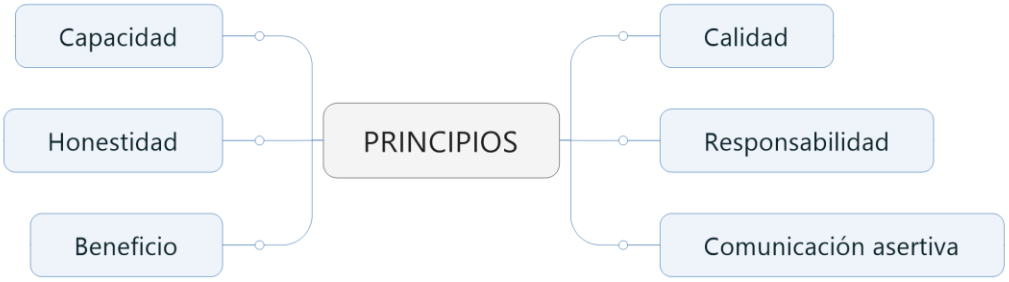
Fase 5 Despliegue del producto: El desarrollador inicia las actividades pertinentes para poner en funcionamiento el producto terminado en el escenario real en el cual debe emplearse, teniendo presentes todas las actividades de origen técnico a que hubiere lugar. Una vez instalado el producto deberá someterse a pruebas en un entorno de producción real.

Fase 6 Cierre del proyecto: como actividades finales del objeto contractual se debe realizar de ser requerido procesos de capacitación en el manejo del producto entregado, así mismos debe dejarse constancia de las personas que en dicha actividad intervienen; generando finalmente un acta de entrega que da fe del recibido a satisfacción por parte del cliente.

9.1.1 Principios

Además de los principios del manifiesto ágil el proceso propuesto acoge los siguientes principios como parte integral para su correcta implementación:

Gráfica 34. Principios del proceso de desarrollo unipersonal propuesto



Fuente: elaboración propia

1. Calidad: el profesional debe asumir su trabajo empleando para ello los más altos estándares de calidad, procurando que el resultado final sea un trabajo de calidad, garantizando siempre la satisfacción del cliente.
2. Responsabilidad: el profesional debe ser responsable de su trabajo en todas las etapas planificadas para lograr el resultado previsto en el alcance consensuado con el cliente, procurando dar cumplimiento oportuno a los cronogramas establecidos.
3. Comunicación asertiva: el profesional debe establecer canales de comunicación que garanticen asertividad, procurando siempre el entendimiento e involucramiento del cliente en los avances, limitaciones y apoyos necesarios para continuar con el proyecto.
4. Capacidad: el profesional ha de ser honesto consigo mismo y con el cliente en relación con su disponibilidad de tiempo para atender el proyecto, por lo tanto, no debe comprometerse a desarrollar trabajos que sobrecarguen su disponibilidad de tiempo y que pueden afectar negativamente la calidad y responsabilidad del profesional.
5. Honestidad: el profesional ha de garantizar siempre la verdad y transparencia con el cliente, procurando no aprovecharse del desconocimiento de la disciplina de este último para beneficio propio.
6. Beneficio: el profesional ha de emprender su trabajo, buscando siempre el mayor beneficio para el cliente, brindando orientación oportuna y clara para que este tome las decisiones pertinentes en equipo con el profesional.

9.1.2 Mejores prácticas del proceso de desarrollo unipersonal propuesto

El proceso de desarrollo unipersonal propuesto adopta las siguientes mejores prácticas.

Tabla 13. Mejores prácticas adoptadas proceso de desarrollo unipersonal propuesto

Prácticas	Descripción
Historias de usuarios	Escritas por el cliente, en su lenguaje, descripciones cortas de lo que el sistema debe realizar. Deben tener el detalle mínimo para estimar tiempo de desarrollo. En la implementación se conocen los detalles. Deben ser programadas entre 1 y 3 semanas; si la estimación es mayor se divide, menor se combina.
Plan de entregas	El cronograma de entregas se desarrolla entre todas las partes interesadas y establece cuáles historias de usuario serán agrupadas para una entrega y su orden. Se desarrolla según las estimaciones de tiempo de desarrollo.
Plan de iteraciones	Las historias de usuario seleccionadas son desarrolladas y aprobadas en un ciclo de iteración. Cada historia de usuario es una tarea específica de programación y establece pruebas de aceptación, las pruebas se hacen al final de cada ciclo de iteración. Define lo que se va a hacer en la iteración y cómo hacerlo, objetivo y propósito.
Recodificación	Escribir nuevamente parte del código de un programa, para corregir errores en el momento en que se introducen aumentando la calidad del software.
Disponibilidad del cliente	Tener el cliente disponible durante todo el proyecto. Los detalles de las historias de usuario son proporcionados por el cliente. Puede prevenir a tiempo situaciones no deseables o funcionamientos inadecuados
Integraciones permanentes	Identifica y corrige problemas de integración cuando son más fáciles de corregir, reduciendo el esfuerzo general mediante una integración frecuente, corregir los problemas inmediatamente a su identificación
Ritmo sostenido	Planificar el trabajo sosteniendo un ritmo constante y razonable, el trabajo extra desmotiva e impacta la calidad del producto.
Pruebas unitarias	Todos los módulos deben pasar las pruebas unitarias antes de ser liberados
Pruebas de detección y corrección	Los errores se corrigen inmediatamente, se toman acciones para que no vuelvan a ocurrir

Pruebas de aceptación	El cliente especifica los escenarios para pruebas. Se crean las pruebas en cada ciclo de iteración. El cliente verifica si los resultados son los adecuados
Iteración	Garantiza la transparencia, inspección y adaptación, habilita la predictibilidad, su resultado es un incremento, durante esta no se admiten cambios que afecten el objetivo de la iteración
Control de Cambios	Debe haber algún proceso para rastrear solicitudes con relación a defectos o requisitos, identificando el producto liberado a través de versiones
Uso de estándares	Programación basada en estándar facilitando el entendimiento y la recodificación
Documentación y capacitación	Reforzar el producto documentando los aspectos claves para su uso a través de sistemas de ayuda y proporcionando capacitación a los usuarios finales
Diseño conceptual	Se genera un diseño aproximado de cómo se vería el producto con lo que hasta el momento se conoce

Fuente: elaboración propia

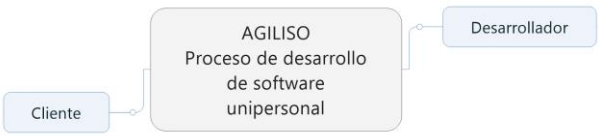
9.1.3 Roles

El proceso de desarrollo unipersonal propuesto identifica dos roles:

Cliente: individuo u organización que esta activamente involucrado en el desarrollo del producto, y cuyos intereses pueden verse afectados positiva o negativamente durante la prestación del servicio del desarrollador, por lo tanto, debe proveer los recursos apropiados y necesarios para una gestión adecuada del proceso de desarrollo.

Desarrollador: encargado de todas las fases del proyecto de desarrollo, liderando siempre en comunicación permanente con el cliente.

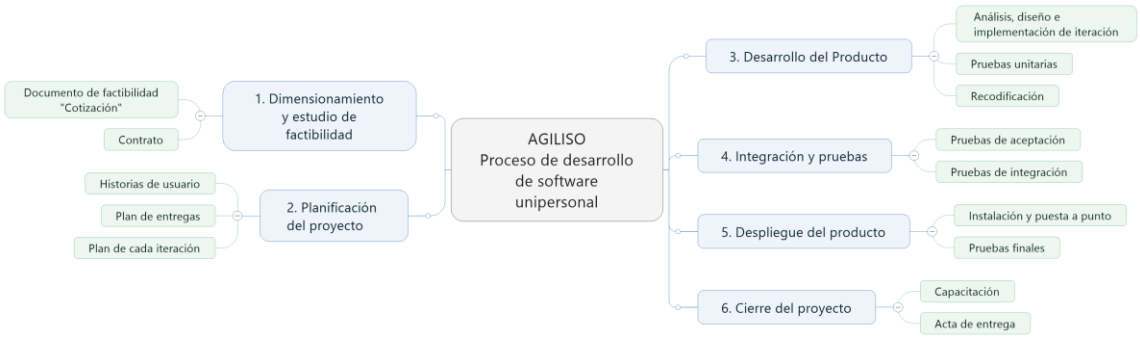
Gráfica 35. Roles proceso de desarrollo unipersonal propuesto



Fuente: elaboración propia

10 ESQUEMA DOCUMENTAL BASE DEL PROCESO DE DESARROLLO UNIPERSONAL PROPUESTO

Gráfica 36. Esquema documental base proceso de desarrollo de software unipersonal propuesto



Fuente: elaboración propia

10.1.1 Documentación fase 1 "dimensionamiento y estudio de factibilidad"

Conlleva la elaboración de los siguientes documentos:

- **Estudio de factibilidad y cotización de servicios de desarrollo de software**: el desarrollador empleará este formato con el objetivo de tener elementos que le permitan estimar costo, esfuerzo y tiempo para el proyecto que se desea iniciar. Se incluye en este documento la siguiente información:
    - Nombre del proyecto: asignar un nombre al proyecto de desarrollo.
    - Duración del proyecto: fecha prevista de inicio y fin del proyecto.
    - Alcance del proyecto: descripción detallada de las funcionalidades y características a ser desarrolladas e implementadas por el desarrollador, este apartado debe ser lo más claro y preciso posible evitando en todo momento ambigüedades sobre el alcance del trabajo a desarrollar por el profesional.
    - Objetivos del proyecto: resultados que se esperan alcanzar con el desarrollo del producto.

- Diagrama conceptual o vista de flujo del sistema: concepción lógica que tiene el desarrollador del trabajo a contratar, presentando a manera de flujo de información o de manera gráfica si visión del desarrollo a contratar.
- Riesgos que pudieran afectar el éxito del proyecto: lista de riesgos identificados que pueden afectar negativamente el desarrollo del proyecto en los tiempos planteados por las partes. Es importante identificar el impacto por la probabilidad de ocurrencia del riesgo de acuerdo con la gráfica 37.

Gráfica 37.Determinación del riesgo probabilidad por impacto

		IMPACTO		
		Bajo	Medio	Alto
PROBABILIDAD	Baja	Muy bajo	Bajo	Medio
	Media	Bajo	Medio	Alto
	Alta	Medio	Alto	Muy alto

Fuente: elaboración propia

- Indique si el proyecto promueve la interoperabilidad de diferentes sistemas y/o bases de datos identificándolos: mencione si el desarrollo que se desea iniciar tendrá dentro de sus funcionalidades interoperabilidad con otro tipo de software.
- Explique la problemática y/o necesidad que será resuelta con el desarrollo del proyecto: describa la problemática y/o necesidad que será resuelta con el desarrollo, tal como el cliente la ha indicado y como ha sido comprendida.
- Indique si para el desarrollo del proyecto se requieren contrataciones adicionales de servicios para cumplir con los objetivos propuestos: mencione si para lograr el desarrollo e implementación exitosa del proyecto el cliente debe realizar contrataciones de servicios adicionales, tales como comunicaciones, instalación de redes, etc.
- Consideraciones de software: mencione las herramientas de software que serán empleadas en el proceso de desarrollo, justificando su selección.
- Consideraciones de hardware: mencione los dispositivos de hardware que deben tenerse en cuenta para garantizar que el proceso de desarrollo e implementación son exitosos.

- Factibilidad técnica: determine la factibilidad técnica de realizar el desarrollo del proyecto en las condiciones específicas requeridas por el cliente, teniendo en cuenta la infraestructura existente y/o la necesidad de nuevas adquisiciones y/o actualizaciones.
- Factibilidad económica: refleje el costo total estimado del proceso de desarrollo que se está cotizando, teniendo en cuenta para ello los costos generales (transportes, papelería, impresiones, etc.), infraestructura (hardware requerido para el desarrollo del proyecto), operativos (costo hora/desarrollador de acuerdo con el tiempo de desarrollo estimado)
- Factibilidad operativa: verificación de la factibilidad en términos de disponer de los recursos humanos necesarios para el desarrollo del sistema propuesto.
- Cronograma de actividades estimadas para el desarrollo del proyecto: identifique de manera general las actividades que serán realizadas durante el desarrollo del proyecto, dando una estimación aproximada de su duración.
- Conclusiones estudio de factibilidad: genere las conclusiones del estudio de factibilidad incluyendo la cotización de los servicios ofrecidos como desarrollador.
    - **Contrato**: Documento legal que determina las cláusulas y condiciones para adelantar el desarrollo de software.

La alineación de los documentos de la fase con la norma ISO 29110 puede apreciarse en la siguiente tabla:

Tabla 14. Alineación de los documentos de la fase 1 con la norma ISO 29110:2014

Documento	Actividad ISO 29110:2014	Objetivos	Proporciona
Estudio de factibilidad y cotización de servicios de desarrollo de software	GP.1	GP.01 GP.05 GP.06 GP.07	El enunciado de trabajo revisado y las tareas necesarias para proveer los entregables necesarios y satisfacer los requisitos del cliente
			El ciclo de vida del proyecto, incluyendo la dependencia de las tareas y su duración
			Los roles y responsabilidades del equipo de trabajo y del cliente

Contrato			Las necesidades de entrenamiento y recursos para el proyecto
			La estimación de esfuerzo, costo y cronograma
			La identificación de los riesgos del proyecto
			La estrategia para el control de versiones y a línea base del proyecto
	IS.1	IS.01	El compromiso por parte del equipo de trabajo y el gestor del proyecto con el plan del proyecto.

Fuente: elaboración propia

10.1.2 Documentación fase 2 "Planificación del proyecto"

Conlleva la elaboración de los siguientes documentos:

- **Historias de usuario**: contiene las descripciones de los requerimientos del cliente, a través de la construcción de un entendimiento de las partes basado en la colaboración, el documento contiene los siguientes campos:
    - Historia de usuario #: asigna un identificador único a la historia de usuario
    - Enunciado de la historia
        - Rol: papel desempeñado por la persona que escribe la historia de usuario.
        - Característica/funcionalidad: necesidad o funcionalidad requerida por el individuo.
        - Razón/resultado: Motivación de la característica/funcionalidad expresada, justificación de la necesidad manifestada.
    - Criterio de aceptación
        - Escenario #: identificación numérica consecutiva (1, 2, 4), de los escenarios identificados para la historia de usuario.
        - Criterio de aceptación (Título): especificación de la característica que permite aceptar el escenario propuesto para la historia de usuario.

        - Contexto: casos en los cuales se puede generar el escenario propuesto.
        - Entradas de datos: datos que se ingresan en el escenario propuesto.
        - Salidas de datos: salida de información generada por la ejecución del escenario.
        - Evento: situación o acción que ocurre como consecuencia de la ejecución del escenario.
        - Resultado/comportamiento esperado: resultado esperado luego de la ejecución del escenario.

Es importante tener presente que se recomienda que para cada historia de usuario se genere máximo un total de cuatro criterios de aceptación. Así mismo, es importante notar que los rótulos "Rol", "Descripción de la funcionalidad", "Razón / resultado" se reemplazan por el contenido de la historia, y los rótulos "Título", "Contexto", "Evento", "Resultado" se reemplazan por el contenido de los criterios de aceptación

- **Plan de entregas**: registra la planificación de las entregas que se realizarán en las diversas iteraciones llevadas a cabo durante el proceso de desarrollo, se tienen en cuenta los siguientes campos:
    - Iteración: número que identifica la iteración en la cual será atendida la historia de usuario.
    - Historias de usuario #: identifica la historia de usuario que será planificada para entrega
    - Dependencia: dependencia de la historia con relación a las demás historias.
    - Prioridad: se priorizan las historias de usuario, el proceso de desarrollo unipersonal recomienda emplear para dicha actividad la técnica de priorización de MoSCoW, sin embargo, el desarrollador puede emplear una técnica diferente de acuerdo con su habilidad y experiencia.

- Técnica de priorización de MoSCoW: define los siguientes cuatro niveles de prioridad para las historias de usuario:
    - **M** - Must have – Debe tener: identifica aquellas historias de usuario que son vitales para el éxito del proyecto.
    - **S-** Should have – Debería tener: identifica aquellas historias de usuario que revisten una funcionalidad importante pero no necesaria o que puede ser satisfecha de otra forma.
    - **C-**Could have – Podría tener: historia de usuario que incluye una funcionalidad que podría tener el producto siempre y cuando el tiempo y los recursos lo permiten.
    - **W-**Won't have – No se incluirá en esta versión: historia de usuario que contiene funcionalidades que no serán incluidas en la versión pero que pueden desarrollarse en una nueva versión.

- Fecha de inicio: fecha en la cual inicia el desarrollo de la iteración.
- Fecha final: fecha programada para finalizar la iteración.
- Duración: Duración total planificado para el desarrollo de la iteración.
- Estado: presenta el estado en el cual se encuentra la historia de usuario, el cual se actualiza en la medida que se van ejecutando las iteraciones los cuales pueden ser:
    - Sin iniciar: el desarrollo de la historia de usuario aún no ha iniciado.
    - Desarrollo: se ha dado inicio al desarrollo de la historia de usuario, pero aún no se ha finalizado.
    - Pruebas: se están realizando las pruebas al desarrollo de la historia de usuario para verificar la calidad del producto.

        - Finalizado: el desarrollo de la historia de usuario ha pasado satisfactoriamente las pruebas y es posible continuar con la planificación del plan de entregas.

- **Plan de cada iteración**: define detalladamente para cada una de las iteraciones que serán realizadas un conjunto de tareas, actividades y recursos. Cada iteración tiene un conjunto de objetivos los cuales se emplean como referencia de evaluación de los resultados obtenidos en la iteración. Este documento contempla los siguientes elementos:
    - Introducción
        - Alcance: identifica claramente los productos que serán generados como resultado de la iteración
        - Referencias: identifica los documentos que deben tenerse en cuenta durante la iteración
    - Planificación
        - Historia de usuario: identificador de la historia de usuario que sea desarrollada en la iteración
        - Tarea: actividad que debe ser realizada para cumplir con los requerimientos especificados en la historia de usuario
        - Duración: duración en días estimada de la actividad
        - Estado: se sugiere tener en cuenta los siguientes estados para las tareas: Sin iniciar, En desarrollo, Terminada
    - Recursos
        - Humanos: identifica los recursos humanos del lado del cliente que son requeridos para el desarrollo de la iteración
        - Financieros: identifica los recursos financieros propios de la iteración requeridos para el desarrollo de esta, tales como desplazamientos, dispositivos, etc.

        - Hardware: identifica los recursos de hardware que son requeridos para el desarrollo de la iteración
        - Software: identifica los recursos de software que son requeridos para el desarrollo de la iteración
        - Otros: identifica otro tipo de recursos requeridos para el desarrollo de la iteración que no es posible categorizar en los anteriores tipos
    - Riesgos que pueden afectar la iteración
        - Descripción del riesgo o problema
        - Probabilidad de ocurrencia: baja, media, alta
        - Impacto en la iteración: bajo, medio, alto
    - Criterios de evaluación: identifica claramente las entregas que deben conseguirse al finalizar la iteración
    - Pruebas/evaluación
        - Fecha de prueba: identifica la fecha en la cual se lleva a cabo la prueba que verifica la conformidad de la iteración con los criterios de evaluación establecidos
        - Objetivo: Objetivo que tiene la prueba realizada
        - Observaciones: descripciones generales de los resultados

Tabla 15. Alineación de los documentos de la fase 2 con la norma ISO 29110:2014

Documento	Actividad ISO 29110:2014	Objetivos	Proporciona
Historias de usuario	GP.1	GP.01 GP.05 GP.06 GP.07	El enunciado de trabajo revisado y las tareas necesarias para proveer los entregables necesarios y satisfacer los requisitos del cliente
Plan de cada iteración			El ciclo de vida del proyecto, incluyendo la dependencia de las tareas y su duración
			La estrategia de aseguramiento de la calidad del proyecto, a través de verificación y validación de los productos de

Plan de entregas			trabajo/entregables, revisiones del equipo de trabajo y del cliente
			La estimación de esfuerzo, costo y cronograma
			La identificación de los riesgos del proyecto
			La estrategia para el control de versiones y la línea base del proyecto
			El Reporte de avance del Proyecto actualizado
		GP.02	Las solicitudes de cambio analizadas y evaluadas del plan con impacto en costos, cronograma y requisitos técnicos
		GP.03	
	GP.2	GP.04	Cambios aprobados en el plan
		GP.05	Las revisiones de los acuerdos con el equipo de trabajo y el cliente
		GP.07	
			El respaldo del repositorio del proyecto y su recuperación en caso de ser necesario
			Evaluar la realización y progreso del plan real contra los objetivos
	GP.3	GP.02	Identificar y evaluar las desviaciones en la realización y problemas en costo, cronograma y técnicos
			Revisar los riesgos del proyecto e identificar nuevos riesgos

Fuente: elaboración propia

10.1.3 Documentación fase 3 “Desarrollo del producto”

Conlleva la elaboración de los siguientes documentos:

- **Análisis, diseño e implementación de iteración**: permite visualizar el diseño general y por iteraciones del producto a desarrollar, contiene los siguientes elementos:
    - Sistema global
        - Arquitectura: permite visualizar la arquitectura general del producto describiendo como ha sido concebida, así mismo se incluye los beneficios y riesgos de esta.

        - Diagrama de componentes: presenta los componentes que conforman el producto y los niveles en los cuales se encuentra cada uno de ellos.
        - Diagrama de despliegue: permite observar la configuración de todos los nodos, mostrando igualmente las especificaciones de cada uno de los nodos.
        - Árbol de navegabilidad: presenta las principales interfaces del producto y su interconexión.
    - Iteración #1
        - Diagrama de actividad: presenta el diagrama de actividades de la iteración.
        - Diagrama de secuencia: presenta el diagrama de secuencias de la iteración.
        - Diagrama de clases: presenta el diagrama de clases de la iteración.

- **Pruebas unitarias**: permite registrar los casos de prueba, realizar seguimiento y documentar los resultados. contiene los siguientes elementos:
    - Propósito: define el propósito del documento de pruebas unitarias
    - Objetivo: lista los elementos, entregables, artefactos y/o documentos que serán objeto de pruebas
    - Referencias: identifica los documentos en los cuales está basado y/o se referencia el documento de pruebas unitarias
    - Caso de prueba: describe el caso de prueba que será aplicado, contiene los siguientes ítems:
        - Componente: identifica el componente que será objeto de prueba
        - Entorno de la prueba: define ambiente de hardware y software para la aplicación de la prueba

        - Caso de prueba (Título): título que identifica el caso de prueba
        - Descripción: descripción del caso de prueba, indicando elementos, funcionalidades y acciones a ser ejercidas en el caso de prueba
        - Funcionalidad/característica: característica o funcionalidad que se está probando
        - Datos/acciones de entrada: descripción de cada entrada requerida para la ejecución de la prueba.
        - Resultado esperado: salida que se espera luego de la ejecución del caso de prueba
        - Procedimientos especiales requeridos: restricción o condicionamiento de la prueba
        - Dependencias con otros casos de prueba: identifica los casos de prueba que deben ser ejecutados antes.
    - Información para el seguimiento
        - Resultado obtenido: informe con los resultados reales arrojados por el caso de prueba
        - Estado: identifica el estado actual del caso de prueba
        - Última fecha de estado: última fecha en la que cambio el estado del caso de prueba o en el que se tomaron acciones referentes al caso de prueba
        - Observaciones: observaciones relacionadas con el caso versus los resultados obtenidos

- **Recodificación**: identifica los eventos en los cuales es necesario realizar un proceso de recodificación del componente debido a pruebas fallidas o con resultados inesperados, para tal fin se emplea el documento "Plan de Cada Iteración", registrando en el campo Observaciones del ítem Evaluación.

- **Otras prácticas**: durante la codificación se debe tener en cuenta seguir los estándares establecidos, y realizar el control de las versiones haciendo uso de un repositorio.

La alineación de los documentos de la fase con la norma ISO 29110 puede apreciarse en la siguiente tabla:

Tabla 16. Alineación de los documentos de la fase 3 con la norma ISO 29110:2014

Documento	Actividad ISO 29110:2014	Objetivos	Proporciona
Análisis, diseño e implementación de iteración Pruebas unitarias Recodificación	GP.2	GP.02 GP.03 GP.04 GP.05 GP.07	El Reporte de avance del Proyecto actualizado
			Las solicitudes de cambio analizadas y evaluadas del plan con impacto en costos, cronograma y requisitos técnicos
			Cambios aprobados en el plan
			Las revisiones de los acuerdos con el equipo de trabajo y el cliente
			El respaldo del repositorio del proyecto y su recuperación en caso de ser necesario
	GP.3	GP.02	Evaluar la realización y progreso del plan real contra los objetivos
			Identificar y evaluar las desviaciones en la realización y problemas en costo, cronograma y técnicos
			Revisar los riesgos del proyecto e identificar nuevos riesgos
			Documentar las solicitudes de cambio, adoptar las acciones correctivas definidas y monitorear los cambios hasta su cierre
	IS.1	IS.01	El establecimiento de un ambiente para la implementación
	IS.2	IS.02 IS.06 IS.07	La obtención, análisis y especificación de los requisitos del cliente
			La verificación y validación de los requisitos
			El control de versiones de los requisitos del producto de software
	IS.3	IS.03	El diseño de la arquitectura el software, los componentes del software y las interfaces asociadas

		IS.06	El diseño detallado de los componentes de software y sus interfaces
		IS.07	La revisión de la especificación de requisitos por parte del equipo de trabajo
			El diseño de software verificado y los defectos corregidos
			La trazabilidad de los requisitos al diseño de software, casos de prueba y procedimientos de prueba
			Productos y documentos de diseño bajo control de versiones
		IS.04	La revisión del diseño de software por parte del equipo de trabajo para determinar la secuencia de construcción del software
	IS.4	IS.06	Los componentes de software codificados y pruebas unitarias aplicadas
		IS.07	La trazabilidad entre los componentes de software y el diseño de software

Fuente: elaboración propia

10.1.4 Documentación fase 4 "Integración y pruebas"

Conlleva la elaboración de los siguientes documentos:

- **Pruebas de integración**: permite registrar los casos de prueba, realizar seguimiento y documentar los resultados. contiene los siguientes elementos:
    - Propósito: define el propósito del documento de pruebas de integración
    - Caso de prueba: describe el caso de prueba que será aplicado, contiene los siguientes ítems:
        - Componentes: identifica los componentes que se integrarán y serán objeto de prueba
        - Entorno de la prueba: define ambiente de hardware y software para la aplicación de la prueba de integración
        - Caso de prueba (Título): título que identifica el caso de prueba

        - Descripción: descripción de lo que será probado en el caso de prueba, indicando elementos, funcionalidades y acciones a ser ejercidas en el caso de prueba
        - Prerrequisitos: prerrequisitos necesarios para ejecutar el caso de prueba de integración de componentes
        - Paso y descripción: consecutivo y descripción de los pasos para implementar el caso de prueba de integración
        - Datos de entrada: descripción de cada entrada requerida para la ejecución de la prueba.
        - Salida esperada: salida que se espera luego de la ejecución del caso de prueba
        - Resultado obtenido: informe con los resultados reales arrojados por el caso de prueba
        - Observaciones: observaciones relacionadas con la ejecución del paso en el caso de prueba de integración versus los resultados obtenidos
    - Información para el seguimiento
        - Estado: identifica el estado actual del caso de prueba
        - Última fecha de estado: última fecha en la que cambio el estado del caso de prueba o en el que se tomaron acciones referentes al caso de prueba
        - Observaciones: observaciones relacionadas con el caso versus los resultados obtenidos
    - Versiones: Consecutivo que identifica de manera única e inequívoca una versión de los componentes liberados para el producto de software
    - Resumen de cambios: descripción general de los cambios más representativos de la versión

    - Fecha: Fecha en la cual se librea la versión.

- **Pruebas de aceptación**: permite registrar los casos de prueba de aceptación, realizar seguimiento y documentar los resultados. contiene los siguientes elementos:
    - Propósito: identifica el propósito general del documento de pruebas de aceptación
    - Alcance: Identifica las personas y/o unidades organizacionales que deben intervenir en las pruebas de aceptación
    - Caso de prueba
        - Descripción: descripción del caso de prueba de aceptación
        - Prerrequisitos: prerrequisitos necesarios para ejecutar el caso de prueba de aceptación
        - Paso y descripción: consecutivo y descripción de los pasos para implementar el caso de prueba de aceptación
        - Datos de entrada: descripción de cada entrada requerida para la ejecución de la prueba.
        - Salida esperada: salida que se espera luego de la ejecución del caso de prueba
        - Resultado obtenido: informe con los resultados reales arrojados por el caso de prueba
        - Observaciones: observaciones relacionadas con la ejecución del paso en el caso de prueba de aceptación versus los resultados obtenidos
    - Información para el seguimiento
        - Estado: identifica el estado actual del caso de prueba

- Última fecha de estado: última fecha en la que cambio el estado del caso de prueba o en el que se tomaron acciones referentes al caso de prueba
- Observaciones: observaciones relacionadas con el caso versus los resultados obtenidos

Se debe generar documentación de manual de usuario y de operación del software dentro del sistema de ayudas del mismo producto de software.

La alineación de los documentos de la fase con la norma ISO 29110 puede apreciarse en la siguiente tabla:

Tabla 17. Alineación de los documentos de la fase 4 con la norma ISO 29110:2014

Documento	Actividad ISO 29110:2014	Objetivos	Proporciona
Pruebas de integración	GP.2	GP.02	El Reporte de avance del Proyecto actualizado
		GP.03	Las solicitudes de cambio analizadas y evaluadas del plan con impacto en costos, cronograma y requisitos técnicos
		GP.04	Cambios aprobados en el plan
		GP.05	Las revisiones de los acuerdos con el equipo de trabajo y el cliente
		GP.07	El respaldo del repositorio del proyecto y su recuperación en caso de ser necesario
Pruebas aceptación	GP.3	GP.02	Evaluar la realización y progreso del plan real contra los objetivos
			Identificar y evaluar las desviaciones en la realización y problemas en costo, cronograma y técnicos
			Documentar las solicitudes de cambio, adoptar las acciones correctivas definidas y monitorear los cambios hasta su cierre
	GP.4	GP.02	La entrega de productos tal como fueron especificados en las instrucciones de entrega
	IS.1	IS.01	El establecimiento de un ambiente para la implementación
	IS.2	IS.02	La verificación y validación de los requisitos del cliente

		IS.06	El control de versiones de los requisitos el producto de software
		IS.07	
	IS.3	IS.03	El diseño de software verificado y los defectos corregidos
		IS.06	Los casos de prueba y procedimientos de prueba verificados para las pruebas de integración
		IS.07	La trazabilidad de los requisitos al diseño de software, casos de prueba y procedimientos de prueba
			Productos y documentos de diseño bajo control de versiones
	IS.5	IS.05	La comprensión de los casos de prueba, procedimientos de prueba y del entorno de integración
		IS.06	Los componentes de software son integrados, los defectos corregidos y los resultados documentados
		IS.07	La trazabilidad de los registros y diseño del producto software integrado
			La documentación y verificación de los manuales de usuario y de operación del software
			El software verificado e incorporado a la línea base

Fuente: elaboración propia

10.1.5 Documentación fase 5 "Despliegue del producto"

Conlleva la elaboración de los siguientes documentos:

- **Instalación y puesta a punto**: constituye las guías que permiten hacer una entrega exitosa del producto al cliente. contiene los siguientes elementos:
    - Fecha: fecha en la cual se hace entrega del producto
    - Identificación de entregables: identifica todos los entregables que constituyen el producto
    - Requerimientos de entrega: identificación de los requerimientos de entrega del producto
    - Orden secuencial de tareas realizadas: organización secuencial de las actividades llevadas a cabo para la instalación y puesta a punto del producto de software

- *Releases* aplicables: versiones aplicables del producto
- Criterios de aceptación: lista los criterios de aceptación del producto de software, así como la fecha de cumplimiento de estos
- Componentes de software: lista los componentes de software del producto entregado, así como la información de su versión
- Respaldo y procedimientos de recuperación: información sobre el respaldo del producto y los procedimientos de recuperación de este

- **Pruebas finales**: permite registrar los casos de prueba final y documentar los resultados. Contiene los siguientes elementos:
  - Propósito: identifica el propósito general del documento de pruebas finales
  - Alcance: Funcionalidades del Producto de software en el entorno de trabajo real del cliente
  - Caso de prueba
    - Descripción: descripción del caso de prueba final
    - Prerrequisitos: prerrequisitos necesarios para ejecutar el caso de prueba final
    - Paso y descripción: consecutivo y descripción de los pasos para implementar el caso de prueba de final
    - Datos de entrada: descripción de cada entrada requerida para la ejecución de la prueba.
    - Salida esperada: salida que se espera luego de la ejecución del caso de prueba
    - Resultado obtenido: informe con los resultados reales arrojados por el caso de prueba

        - Observaciones: observaciones relacionadas con la ejecución del paso en el caso de prueba final versus los resultados obtenidos
        - Fecha: fecha de ejecución de la prueba final
    - Información de resultados
        - Observaciones: observaciones relacionadas con el caso versus los resultados obtenidos

La alineación de los documentos de la fase con la norma ISO 29110 puede apreciarse en la siguiente tabla:

Tabla 18. Alineación de los documentos de la fase 5 con la norma ISO 29110:2014

Documento	Actividad ISO 29110:2014	Objetivos	Proporciona
Instalación y puesta a punto	GP.4	GP.02	El Reporte de avance del Proyecto actualizado
	IS.1	IS.01	El establecimiento de un ambiente para la implementación
Pruebas finales	IS.6	IS.06	El manual de mantenimiento verificado
		IS.07	

Fuente: elaboración propia

10.1.6 Documentación fase 6 "Cierre del proyecto"

Conlleva la elaboración de los siguientes documentos:

- **Plan y registro de capacitación**: permite registrar la planificación y registrar la asistencia a la misma por parte del cliente. Contiene los siguientes elementos:
    - Propósito: Planificar la actividad de capacitación al cliente posterior a la entrega del producto.
    - Objetivos: registra los objetivos que se busca conseguir con la capacitación
    - Contenido
        - No: consecutivo de las sesiones de capacitación planificadas
        - Capacitación: módulo o tema objeto de la capacitación

        - Fecha aproximada. Fecha en la cual se proyecta orientar la capacitación
        - Total, horas: cantidad total de horas invertidas en la capacitación
        - Cantidad. participantes: cantidad total proyectada de asistentes a la capacitación
    - Registro de asistencia
        - Participantes: listado de participantes a las capacitaciones
        - Cap. 1: se marca con una "X" la asistencia del participante a la capacitación 1
        - Fecha: fecha real de la capacitación
        - Firma: firma del participante en el proceso de capacitación
- **Acta de entrega**: documento mediante el cual se formaliza la entrega del producto de software. contiene los siguientes elementos
    - Criterios de aceptación: lista los criterios de aceptación y la fecha en la cual se cumplieron los mismos.
    - Firmas de aceptación: relación de los componentes del producto del software, versión, fecha de aceptación y se formaliza con la firma del cliente.

La alineación de los documentos de la fase con la norma ISO 29110 puede apreciarse en la siguiente tabla:

Tabla 19. Alineación de los documentos de la fase 6 con la norma ISO 29110:2014

Documento	Actividad ISO 29110:2014	Objetivos	Proporciona
Planeación y registro de capacitación	GP.4	GP.02	El Reporte de avance del Proyecto actualizado
			Contar con un soporte de la aceptación del producto por parte del cliente de acuerdo con las instrucciones de entrega
			La culminación del proyecto y firma del acta de aceptación

Acta de entrega	IS.6	IS.06 IS.07	La entrega del producto de software y la documentación aplicable de acuerdo con las instrucciones de entrega

Fuente: elaboración propia

11 HERRAMIENTA PARA SOCIALIZACIÓN DEL PROCESO

Se realizó la investigación de software GNU/GPL disponible que permitiera distribuir y socializar el proceso de desarrollo unipersonal propuesto con la finalidad de realizar las pruebas de concepto, para determinar cuál era la más pertinente, se llevó a cabo un proceso de calificación de acuerdo con los factores de presentados en la tabla 20

Tabla 20. Criterios de calificación de las plataformas GNU/GPL que permiten la socialización del proceso de desarrollo unipersonal propuesto

CRITERIO	VALOR	DESCRIPCIÓN
Actualización	1	No se ha vuelto a actualizar
	2	Actualizado al menos una vez por año
	3	Actualizado al menos dos veces por año
	4	Actualizado más de dos veces al año
Instalación	1	Instalación manual, a través de comandos
	2	Instalación guiada con requerimientos de conocimientos técnicos
	3	Instalación guiada sin requerimientos de conocimientos técnicos
Facilidad de uso	1	Requiere amplia capacitación para su uso
	2	Requiere inducción sencilla para su uso
	3	Intuitivo y fácil de emplear
Soporte	1	No tiene soporte
	2	Soporte algunos días y por correo electrónico
	3	Soporte todos los días chat, teléfono y correo
Administración	1	No posee módulo de administración
	2	Posee módulo de administración, requiere amplio conocimiento
	3	Posee módulo de administración, no requiere amplio conocimiento
Aplicación WEB	1	No posee aplicación WEB
	2	Solo posee aplicación de escritorio con entorno de red
	3	Posee aplicación WEB
Compatibilidad	1	Solo para plataformas Linux
	2	Solo para plataformas Windows

	3	Permite instalación en plataformas Linux y Windows
Tiempo de respuesta	1	Más de 1 minuto
	2	Entre 40 y 60 segundos
	3	Entre 20 y 40 segundos
	4	Entre 5 y 20 segundos
	5	Menos de 5 segundos
Seguridad	1	No posee seguridad
	2	Seguridad básica solo usuario y contraseña
	3	Altos niveles de seguridad, incluye logs y encriptación

Fuente: elaboración propia

Los resultados obtenidos luego del proceso de evaluación se resumen en la gráfica 38:

Gráfica 38. Selección de software GNU/GPL socialización proceso de desarrollo unipersonal

SOFTWARE	VER	ACTUALIZACIÓN	INSTALACIÓN	FACILIDAD DE USO	SOPORTE	ADMINISTRACIÓN	APLICACIÓN WEB	COMPATIBILIDAD	TIEMPO DE RESPUESTA	SEGURIDAD	TOTAL
		1-4	1-3	1-3	1-3	1-3	1-3	1-3	1-5	1-3	
SECURIA SGSI	1.2.4	2	3	1	2	2	1	2	3	2	18
ERAMBA	Community	3	1	1	2	2	3	1	4	3	20
EPF - COMPOSER	1.5.1.8	2	3	2	3	3	3	3	4	1	24

Fuente: Propia

El puntaje más alto lo obtuvo el software EPF – Composer, por lo que se generó un sitio web con esta herramienta, que facilitó la socialización del proceso de desarrollo de software unipersonal propuesto a los desarrolladores que participaron en las pruebas de concepto.

Gráfica 39. Pantalla principal sitio web para socializar el proceso unipersonal propuesto

Fuente: elaboración propia

Gráfica 40. Descripción de plantillas sitio web socialización del proceso unipersonal propuesto

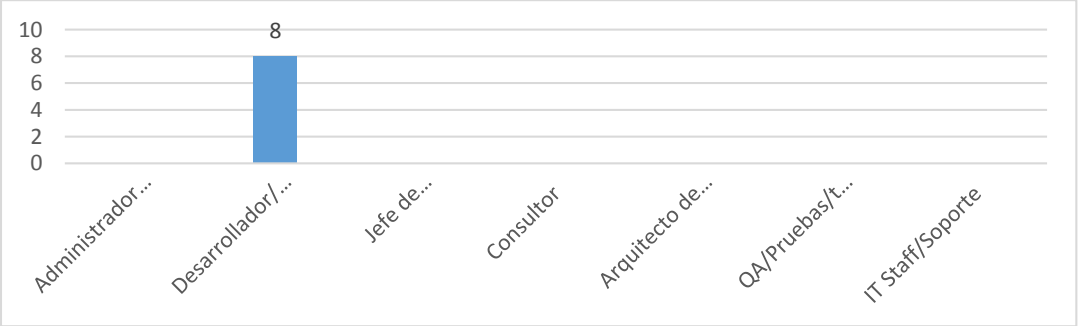
Fuente: elaboración propia

Es importante notar que el sitio web diseñado, les permite a los desarrolladores descargar los artefactos para ser empleados y ajustados de acuerdo a sus necesidades.

11.1.1 Prueba de concepto

Se socializó el sitio web generado y se acompañó de una encuesta (ver Anexo 1), que fue aplicada a 8 desarrolladores obteniendo los siguientes resultados y observaciones:

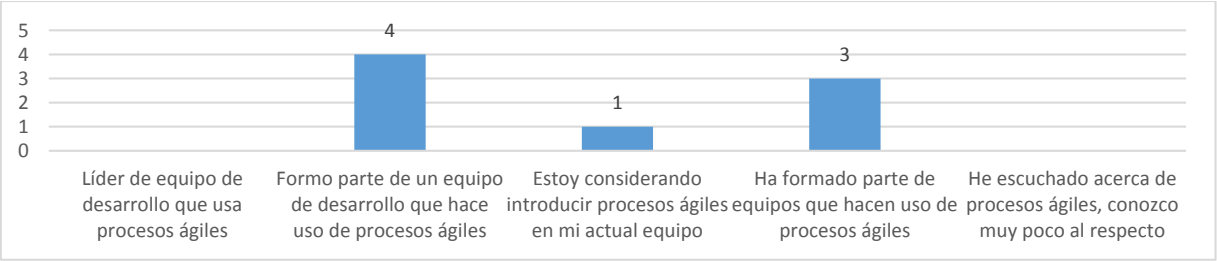
Gráfica 41. ¿Cuál de los siguientes roles define mejor su actual puesto de trabajo?



Fuente: elaboración propia

El 100% de los profesionales que participaron en la prueba de concepto son desarrolladores o programadores en la actualidad.

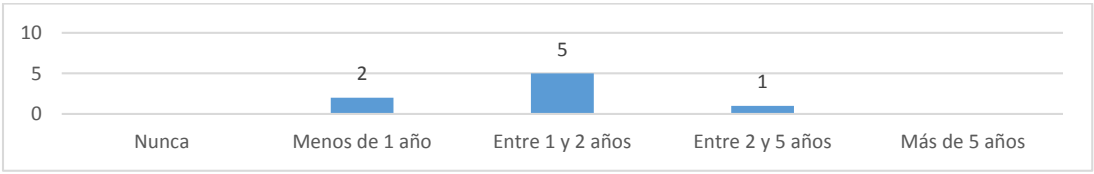
Gráfica 42. ¿Cuáles de las siguientes situaciones describe mejor su nivel de conocimiento en procesos ágiles?



Fuente: elaboración propia

Del total de profesionales, el 50% forman parte de un equipo de desarrollo que hace uso de procesos ágiles, entre tanto el 12.5% están considerando introducir procesos ágiles en el equipo de desarrollo actual y el 37,5% han formado parte de equipos que hacen uso de procesos ágiles, lo que refleja la idoneidad de los profesionales para el desarrollo de la prueba de concepto.

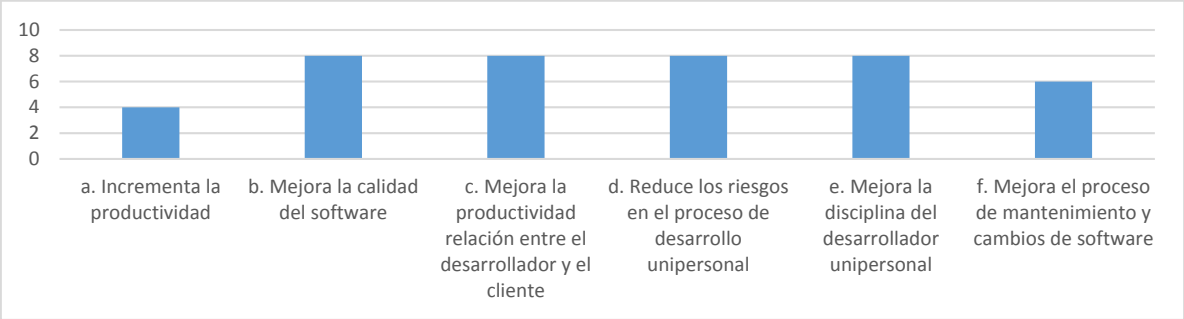
Gráfica 43. ¿Por cuánto tiempo ha hecho uso personalmente de metodologías ágiles sin tener en cuenta desarrollos unipersonales o en equipos de desarrollo?



Fuente: elaboración propia

El 25% de los desarrolladores encuestados han hecho uso personalmente de procesos ágiles por un periodo de tiempo inferior a 1 año, mientras que el 62,5% los han utilizado entre 1 y 2 años y el 12,5% entre 2 y 5 años, lo cual refleja que los desarrolladores que participaron en la prueba de concepto han empleado en su actividad profesional procesos ágiles, convirtiéndose esta característica en un referente para el proceso de desarrollo unipersonal propuesto.

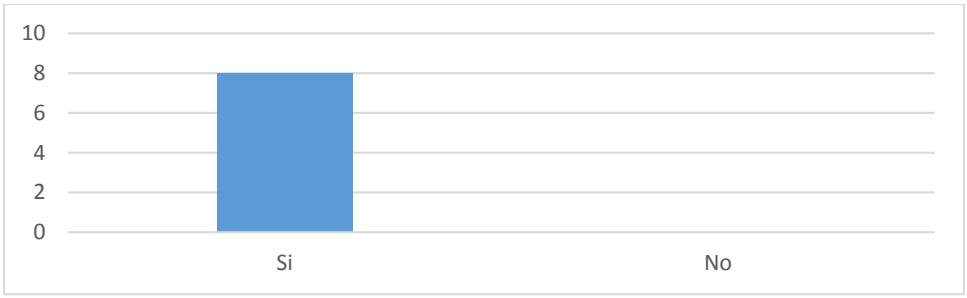
Gráfica 44. Desde su perspectiva y experiencia ¿cuáles considera usted pueden ser los beneficios del proceso de desarrollo unipersonal propuesto?



Fuente: elaboración propia

Es interesante notar que todas las personas que fueron consultadas en la prueba de concepto determinan que la adopción de un proceso desarrollo unipersonal genera beneficios tanto para el desarrollador como para el cliente, siendo las preguntas b, c, d y e categorizadas como beneficios del proceso de desarrollo unipersonal por el 100% de los profesionales, mientras que el 75% indica f como beneficio y el 50% indican la opción a.

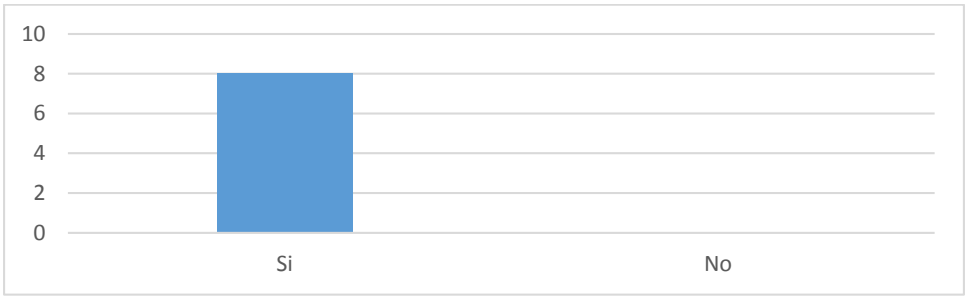
Gráfica 45. ¿El sitio web muestra claramente el proceso de desarrollo, describiendo cada una de las etapas que contempla?



Fuente: elaboración propia

El 100% de los desarrolladores coinciden en afirmar que el sitio web con el proceso de desarrollo unipersonal propuesto describe de manera clara cada una de las etapas que contempla.

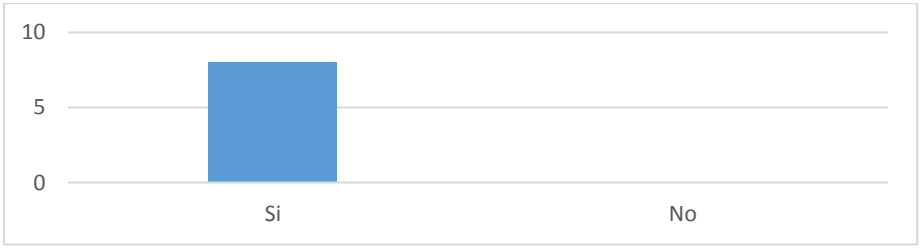
Gráfica 46. ¿Considera que se conserva la agilidad en el proceso de desarrollo unipersonal propuesto formalizándolo mediante el esquema documental propuesto?



Fuente: elaboración propia

El 100% de los desarrolladores afirman que no se ve afectada la agilidad mediante la aplicación de la documentación que acompaña al proceso de desarrollo unipersonal propuesto.

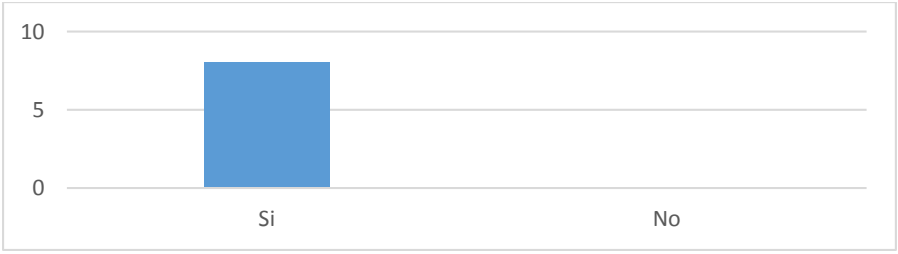
Gráfica 47. ¿El sitio web presentado es fácil de entender?



Fuente: elaboración propia

El 100% de los desarrolladores no tuvieron problemas en comprender como aplicar el proceso de desarrollo unipersonal expuesto en el sitio web, informando que su estructura y etapas están claramente identificadas y documentadas, facilitando su seguimiento.

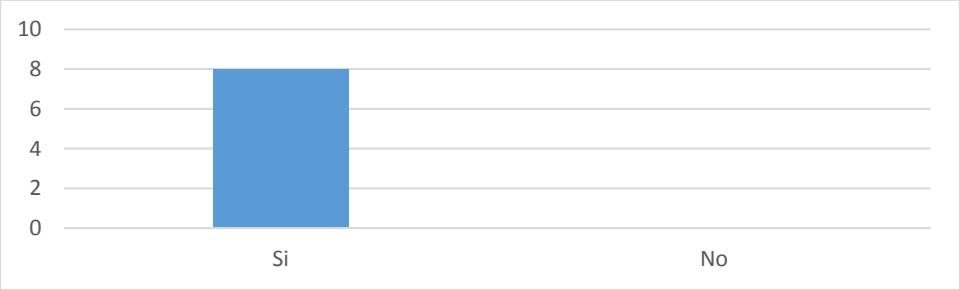
Gráfica 48. ¿El sitio web identifica de manera clara los roles de las personas que intervienen en el proceso de desarrollo unipersonal?



Fuente: elaboración propia

El 100% de los desarrolladores, identificaron rápidamente los roles de las personas que intervienen en el proceso de desarrollo unipersonal propuesto, así mismo indican la facilidad de indagar sobre cada uno de ellos y sus tareas.

Gráfica 49. ¿El diagrama del proceso es fácil de comprender y muestra su totalidad?



Fuente: elaboración propia

El 100% de los desarrolladores indican que el proceso es claro, pertinente y que cada una de las fases está claramente identificada y explicada, conducentes a una guía detallada de la aplicación del proceso de desarrollo unipersonal.

Dentro de las observaciones finales realizadas en la prueba de concepto, se identificó que es importante no solo publicar el formato o plantilla de cada artefacto en cada una de las fases, sino que a manera de guía es igualmente importante publicar en el sitio web un ejemplo descriptivo de cada uno de ellos.

11.1.2 Prueba de valor

Se definió como piloto un caso de uso consistente en el desarrollo de un módulo de aplicación de escritorio para control de parqueaderos, incluyendo el manejo tarifario, clasificación vehicular; el lenguaje de programación en el cual se solicitó el producto fue C# y se planeó en total 1 iteración, el producto fue diseñado y desarrollado en un contexto académico.

Esta prueba se aplicó a dos desarrolladores con el mismo piloto de desarrollo de software, uno de los cuales empleó su proceso habitual y el otro el proceso propuesto, una vez finalizadas las pruebas se aplicaron las métricas de: esfuerzo, cubrimiento de requisitos, terminación de módulos en tiempos esperados y duración del proyecto, arrojando los siguientes resultados (ver tabla 21):

Tabla 21. Pruebas de valor

Métricas	Con proceso	Sin proceso
Esfuerzo	42 h/h	30 h/h
Cubrimiento de requisitos	100 %	100 %
Terminación de módulos en los tiempos esperados	100 %	100 %
Duración del proyecto	7 días	5 días

Fuente: elaboración propia

11.1.2.1 Esfuerzo

Se puede observar que el desarrollador que aplicó el proceso propuesto realizó un esfuerzo mayor puesto que debió respetar cada uno de los elementos propuestos en cada una de las diferentes fases. Por otro lado, el desarrollador quien realizó el piloto bajo su propio modelo, lo hizo invirtiendo menos tiempo, teniendo en cuenta que son actividades que realiza continuamente y que han sido definidas según sus propias necesidades.

11.1.2.2 Cubrimiento de requisitos

Los dos desarrolladores cumplieron el 100% de los requisitos y no manifestaron dudas respecto a las necesidades planteadas dentro de éstos.

11.1.2.3 Terminación de módulos en tiempos esperados

Los dos desarrolladores hicieron entrega del trabajo solicitado, dentro de los tiempos estimados para el piloto.

11.1.2.4 Duración del proyecto

Teniendo en cuenta el esfuerzo, podemos ver una coherencia con respecto a la duración del proyecto por parte de los dos desarrolladores. El desarrollador que aplicó el proceso invirtió 7 días en ejecutar el piloto propuesto, mientras que el desarrollador quien trabajó bajo su propio proceso invirtió 5 días para cumplir con el mismo ejercicio, siendo esta una diferencia de 2 días respecto del primero.

11.1.3 Prueba de uso

Con el objetivo de determinar si los resultados de la aplicación del proceso propuesto son los esperados se emplearon las siguientes métricas: cantidad de defectos, costos de desarrollo, fallas de requerimientos descubiertas, cantidad de inconformidades relacionadas con el proceso, puntos de historia de usuario por iteración y líneas de código, ver tabla 22

Tabla 22 Resultados pruebas de uso

Métrica	Con proceso	Sin proceso
Cantidad de defectos	7	sin documento
Costo de desarrollo	$472.500	$337.500
Fallas de requerimientos descubiertos	0	0
Cantidad de inconformidades relacionadas con el proceso	1	N/A
Puntos de historia de usuario por iteración	4	N/A
Líneas de código apróximadas	800	900
Cantidad de documentos de soporte que se entregan	14	4

Fuente: elaboración propia

11.1.3.1 Cantidad de defectos

El desarrollador quién trabajo bajo el proceso propuesto, realizando las actividades pertinentes referentes a las pruebas, logró identificar 7 defectos durante las iteraciones y las fases finales, mientras que el desarrollador que trabajó bajo su propio proceso no documentó la cantidad de defectos encontrados.

11.1.3.2 Costos de desarrollo

Tomando como base un salario de $1’800.000 para un desarrollador en la ciudad de Manizales, se logró establecer que el costo por ejecutar el piloto fue de $472.500 por parte del desarrollador que trabajó bajo el proceso propuesto y $337.500 por parte del desarrollador que aplicó el piloto bajo su propio proceso.

11.1.3.3 Fallas de requerimientos descubiertas

Ninguno de los desarrolladores encontró fallas de requerimientos.

11.1.3.4 Cantidad de inconformidades relacionadas con el proceso

Esta métrica aplica únicamente para el desarrollador que aplicó el proceso propuesto, y refleja que tuvo una inconformidad, principalmente con la documentación, debido a que, dentro de sus expectativas al implementar el proceso, no contaba con la carga documental que tuvo que realizar pues está acostumbrado a generar aplicaciones con un enfoque de aplicación operativa y funcional a través de un desarrollo basado en la experiencia, no guiado y documentado.

11.1.3.5 Puntos de historia de usuario por iteración

Esta métrica aplica únicamente para el desarrollador que aplicó el modelo propuesto. La métrica nos dice que se implementaron 4 historia de usuario por iteración. Esto es teniendo en cuenta que es el primer acercamiento del usuario con el proceso.

11.1.3.6 Líneas de código

El número de líneas de código aproximado es muy cercado, siendo 800 líneas para el desarrollador que implementó el piloto bajo el proceso propuesto y 900 líneas de código para el desarrollador que lo hizo bajo su propio proceso.

11.1.3.7 Cantidad de documentos de soporte que se entregan

El desarrollador que aplicó el proceso propuesto generó en total 14 documentos que acompañaron el desarrollo de la iteración constituyendo un instrumento valioso para futuras modificaciones y/o ampliaciones de alcance del producto de software, por parte de él mismo o de otro desarrollador. Entre tanto, el desarrollador que no empleó el proceso propuesto entrego 4 documentos (contrato, acta de entrega, manual de usuario, manual técnico), por lo cual no es posible realizar trazabilidad al proceso de desarrollo.

12 DISCUSIÓN DE RESULTADOS

La implementación de los requisitos propuestos en el caso de uso se llevó a cabo aplicando el proceso de desarrollo unipersonal propuesto (AGILISO), por parte de un egresado de programación, quien manifestó la importancia de contar con una guía que acompañe al desarrollador desde el principio, principalmente con los acuerdos contractuales con el cliente. Esta etapa es bastante importante porque define las condiciones bajo las cuales se desarrollará el proyecto además de establecer el acuerdo de trabajo según los compromisos establecidos y la capacidad a nivel técnico y carga laboral que pueda disponer el desarrollador para realizar un trabajo de calidad.

El proceso requiere de tiempo para concluir su aprendizaje, es por esto, que la métrica de los tiempos obtenidos durante el piloto arroja resultados diferentes; el desarrollador que aplicó el proceso propuesto tuvo que apropiarse de nuevos conceptos y técnicas para cumplir en cada etapa, con los objetivos definidos.

Uno de los valores que se establece como relevante dentro del proceso es la disciplina. Si bien los dos desarrolladores manifestaron haber trabajado de manera continua para cumplir con los requisitos, se pudo notar que la persona que aplicó el proceso lo hizo de manera constante, secuencial y ordenada, asignando tiempos y espacios de trabajo como es debido, mientras que el segundo desarrollador quién realizó la implementación bajo su propio proceso, lo hizo bajo una estrategia que no definía periodos de trabajo organizados, es decir, sin horarios y espacios establecidos, lo que afectaba la verificación de los avances establecidos en su planeación. Claro está que tenía periodos de trabajo muy intensos que, sumados al conocimiento previo y su dominio sobre su propia técnica, le permitían acelerar las entregas.

En cuanto a las pruebas, el desarrollador quién aplicó el proceso logró identificar y corregir varios defectos durante las iteraciones, mientras que el segundo desarrollador no documentó hallazgo alguno. En este segundo caso, analizando el código fuente entregado, logramos identificar al menos 3 defectos que lastimosamente llegaron al cliente, lo que implica un sobrecosto bastante alto teniendo en cuenta que ahora el cliente deberá incurrir en reprocesos para corregirlo y esperando que dichos defectos no afecten otros componentes del sistema.

Se pudo notar que el entrenamiento específico en cualquier tipo de proceso de desarrollo de software influye en el adecuado seguimiento de las fases propuestas en el proceso AGILISO, así como en el desempeño específico de cada una de las etapas, siendo más notorio en las fases iniciales del proceso propuesto debido a la formalidad de estas.

Tabla 23 Comparación entre las mejores prácticas entre los procesos ágiles analizados, ISO 29110:2014 y AGILISO

N	Proceso -Norma	XP	Scrum	OpenUp	Kanban	PSP	ISO 29110:2014		AGILISO
	Práctica	A	A	A	A	A	A	Obs	A
1	Historias de usuarios	X	X	X			X	IS.2	X
2	Plan de entregas	X	X	X	X	X	X	GP.1 GP.3 GP.4	X
3	Plan de iteraciones	X	X	X		X	X	GP.1 IS.4	X
4	Reuniones diarias de seguimiento	X	X	X	X		X	GP.2 GP.3	
5	Soluciones “Spike”	X							
6	Recodificación	X		X		X	X	IS.4	X
7	Metáforas	X							
8	Disponibilidad del cliente	X	X	X	X		X	GP.2 GP.4	X
9	Uso de estándares	X				X	X	IS.3	X
10	Programación dirigida por pruebas	X		X			X	IS.3 IS.4	
11	Programación en pares	X							
12	Integraciones permanentes	X	X	X	X	X	X	IS.3	X
13	Propiedad del código	X							
14	Ritmo sostenido	X	X		X	X	X	GP.1 GP.3	X
15	Pruebas unitarias	X	X	X	X	X	X	IS.4 IS.5	X
16	Pruebas de detección y corrección	X	X	X	X	X	X	GP.4 IS.4 IS.5	X
17	Pruebas de aceptación	X	X	X	X	X	X	GP.4 IS.6	X
18	Propietario del producto		X						
19	Equipo de desarrollo		X	X					

20	Scrum Máster		X		X					
21	Sprint		X	X		X	X	IS.4		
22	Transparencia		X	X	X					
23	Inspección		X	X	X	X	X	GP.3		
24	Compromiso	X	X	X	X	X				
25	Foco		X				X	IS.1		
26	Tamaño equipo desarrollo		X	X			X	GP.1		
27	Autogestión		X	X						
28	Autosuficiencia		X	X						
29	Sub equipos		X							
30	Casos de Uso			X			X	IS.2		
31	Control de Cambios	X	X	X	X	X	X	GP.2 GP.3	X	
32	Personalización del proceso			X		X	X	IS.3		
33	Documentación y capacitación			X		X	X	IS.6	X	

Fuente: elaboración propia

La cantidad de mejores prácticas adoptadas por el proceso propuesto AGILISO es de 15, de un total de mejores prácticas identificadas de 33, lo que implica un 45,45% de adopción de mejores prácticas; sin embargo, es importante notar que al ser un proceso de desarrollo unipersonal las mejores prácticas orientadas a ser implementadas por equipos de desarrollo no fueron tenidas en cuenta, adoptando un total de 15 mejores prácticas de un total aplicable al desarrollo unipersonal de 23, con un porcentaje de adopción de mejores prácticas identificadas de 65,21%, el 34,78% restante fueron omitidas para no comprometer la agilidad del proceso propuesto.

13 CONCLUSIONES

- La documentación del software no puede ser evitada sino más bien adaptada a una filosofía ágil. En otras palabras, podemos documentar el código fuente bajo un estándar como el que propone Java, que sirva de ayuda técnica al equipo de desarrollo o posteriores participantes del proyecto. Bajo esta estrategia, el desarrollo unipersonal mantiene su dinamismo y aporta a la consecución de los objetivos del proceso propuesto.
- La aplicación de un proceso para el desarrollo de proyectos de software es más amena, práctica y confiable si tenemos a la mano la información que lo define y la guía de uso como lo es para este caso, el sitio web generado con EPF Composer.
- Basar la implementación de requisitos en historias de usuario simples y claras aplicables en cada iteración, reduce la brecha existente entre una necesidad real y una solución adecuada, teniendo en cuenta que los cambios identificados no afectarán el desarrollo del proyecto puesto que se resolverán de forma inmediata de la mano del cliente.
- Aunque el proceso de software se realice de manera unipersonal, es importante realizar las diferentes etapas de pruebas de la manera en que se propone en el presente proceso, puesto que así se logra identificar algún tipo de defecto en diferentes momentos aplicando diferentes técnicas.
- Al seguir el proceso propuesto, la cantidad de defectos detectados y atendidos oportunamente por el desarrollador permite reducir las inconformidades percibidas por el cliente.
- Generalmente se pasan por alto muchas condiciones laborales al momento de concretar un nuevo proyecto, es por eso que el presente proceso sirve como guía para que desde el comienzo se pueda definir de manera organizada cada uno de los elementos a tener en cuenta, como son, para citar algunos, el cronograma de trabajo, la estimación del costo y el tiempo.
- El compromiso, la responsabilidad y la honestidad son algunos de los valores presentes en la ética de cada profesional, es por eso, que, como valores, el desarrollador debe ser consciente del compromiso que acepta por cada proyecto que inicia. El proceso

propuesto enfatiza en estos y más valores, y guía en pro de la construcción de un producto de calidad.

- La aplicación del proceso de desarrollo unipersonal propuesto en el entorno académico en el cual se empleó ha permitido generar una introspección de conceptos de desarrollo con relación a las metodologías ágiles, que acentúan la importancia de un desarrollo de software ordenado y secuencial, fortaleciendo las habilidades de gestión de proyectos de software.

14 RECOMENDACIONES

- Es importante que las instituciones de formación que desarrollan competencias en programación (a nivel técnico laboral, técnico profesional, tecnólogo, pregrado y postgrado) concienticen a los profesionales en la importancia de aplicar un proceso de desarrollo tanto para él mismo como para el cliente.
- Se debe realizar un pilotaje con un desarrollo más grande y con más número de iteraciones, involucrando más programadores y en un contexto más cercano al real, de tal manera que puedan obtenerse conclusiones más acertadas sobre el proceso de desarrollo unipersonal propuesto.
- Desarrollar una investigación que como resultado provea estrategias que permitan que los desarrolladores unipersonales adopten procesos de desarrollo en procura de generar alta calidad y competitividad en la gestión y desarrollo de este tipo de proyectos software.
- Se debe estudiar el impacto real que ha tenido el desarrollo unipersonal llevado a cabo sin procesos de desarrollo en los diversos sectores industriales, de tal manera que sea posible dimensionar sus implicaciones en términos económicos y operativos, en procura de generar estrategias desde los Ministerios de tecnologías de información y comunicación y Educación con relación a la importancia de permear una cultura de proceso en el desarrollo de software unipersonal.

15 REFERENCIAS BIBLIOGRÁFICAS

ACIS - Asociación Colombiana de Ingenieros de Sistemas. (2016). *La aplicación de la gestión de proyectos como factor diferenciador en proyectos de alto impacto.* Memorias, Asociación Colombiana de Ingenieros de Sistemas, Bogotá, Colombia. Recuperado el 19 de 12 de 2017, de http://acis.org.co/portal/content/xiv-jornada-de-gerencia-de-proyectos

Agarwal, R., & Umphress, D. (2008). Extreme programming for a single person team. *Proceedings of the 46th Annual Southeast Regional Conference*, (págs. 82-87). Auburn, Alabama. doi:10.1145/1593105.1593127

Alvarez M., I. (2007). Desarrollo ágil con Scrum. *SG-07 Conferencia y expo.* México. Obtenido de http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:sg07.p02.scrum.pdf

Bahit, E. (08 de 11 de 2011). *Desarrollo ágil con Kanban*. Recuperado el 19 de 12 de 2017, de https://desarrolloweb.com/articulos/desarrollo-agil-kanban.html

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *Manifiesto Ágel*. Obtenido de http://agilemanifesto.org/iso/es/principles.html

Borja López, Y. (2015). Metodología ágil de desarrollo de software - XP. *Repositorio de la Universidad de las Fuerzas Armadas ESPE*, 9-10. Recuperado el 15 de 12 de 2017, de http://www.runayupay.org/publicaciones/2244_555_COD_18_290814203015.pdf

Brito Abundis, C. J. (2013). Metodologías para desarrollar. *ReCIBE*(3), 1-5. Recuperado el 15 de 10 de 2017

Britto Montoya, J. A. (2014). *Adaptación de un proceso de desarrollo de software basado en buenas prácticas.* Tesis de Maestría, Universidad Autónoma de Manizales, Facultad de ingeniería, Manizales, Caldas.

Bustamante, D., & Rodríguez, J. C. (2014). *Metodología XP*. Universidad experimental de los llanos occidentales Ezequiel Zamora, Barinas, Venezuela. Recuperado el 14 de 12 de 2017, de http://blogs.unellez.edu.ve/dsilva/files/2014/07/Metodologia-XP.pdf

Canós, J., Letelier, P., & Penadés, M. (2004). *Metodologías ágiles en el desarrollo de software.* Universidad Politécnica de Valencia, Sistemas, informática y computación, Valencia, España.

Celis Mendoza, O. B. (03 de 07 de 2014). *Diferencias, ventajas y desventajas estre Scrum, XP, OpenUp e ISO 12207*. Recuperado el 21 de 12 de 2017, de https://prezi.com/vugjhc65whet/diferenecias-ventajas-y-desventajas-entre-scrumxpopenup-y-iso-12207/

Colombani, M. A., Pérez, M. M., & Pacífico, C. D. (2016). *Metodologías para el desarrollo de software en PYMES.* Universidad Nacional de Entre Ríos - UNER, Facultad de ciencias de la administración, Concepción del Uruguay, Entre Ríos, Argentina.

Colombia Dev. (11 de 03 de 2016). *Medium.com*. (M. Cabrera, & D. Przybilla, Editores) Recuperado el 03 de 12 de 2017, de https://medium.com/colombia-dev/an%C3%A1lisis-encuesta-salarios-desarrolladores-colombiadev-2016-9969a621ec39

Eclipse. (30 de 05 de 2012). *epf.eclipse.org*. Recuperado el 20 de 12 de 2017, de http://epf.eclipse.org/wikis/openup/index.htm

Florez Marín, L., & Grisales Tobón, F. (2014). *Formulación de criterios para la selección de metodologías de desarrollo de software.* Universidad Tecnológica de Pereira, Facultad de ingenierias, Pereira, Risaralda. Recuperado el 19 de 12 de 2017, de http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/5120/00512F634.pdf;sequence=1

Garces Guayta, L. R., & Egas, L. M. (2012). *Evolución de las metodologías de desarrollo de la ingeniería de software en el proceso de la ingeniería de software.* Universidad de las fuerzas armadas - ESPE, Sangolquí, Ecuador.

Gimson, L. (2012). *Metodologías ágiles y desarrollo basado en conocimiento.* Tesis, Universidad Nacional de la Plata, Facultad de informática, Argentina. Recuperado el 20 de 12 de 2017, de http://sedici.unlp.edu.ar/bitstream/handle/10915/24942/Documento_completo__.pdf?sequence=1

Grinell, R. M. (2005). *Social work research of evaluation quantitative and qualitative approaches* (6a ed.). (O. U. Press, Ed.) New York, EE.UU.

Grupo ISSI - Ingeniería del Softtware y Sistemas de Información . (2003). Meodologias Ágiles en el desarrollo de software. *VIII Jornadas de Ingeniería de Software y Bases de Datos, JISBD, VIII*, págs. 10-15. Alicante, España.

Helms, Remko & Giovacchini, Elia & Teigland, Robin & Kohler, & Thomas. (2010). A Design Research Approach to Developing User Innovation Workshops in Second Life. *Journal For Virtual Worlds Research*(10), 3.

ICONTEC. (2014). *NTC ISO/IEC TR 29110-5-1-2.* (ICONTEC, Ed.) Bogotá, Colombia.

IEEE Computer Society. (2014). *SWEBOK V.3 Guide to the software engineering body of knowledge* (Vol. 3). Piscataway, New Jersey: IEEE. Obtenido de www.swebok.org.

iso25000.com. (2017). *ISO 25000 Calidad del producto de software*. Recuperado el 30 de 11 de 2017, de http://iso25000.com/index.php/normas-iso-25000/iso-25010?limit=3&limitstart=0

Laporte, C. Y. (2016). La implementación de la norma ISO/IEC 29110 guías de gestión e ingeniería para las organizaciones pequeñas. *Congreso internacional de mejora de procesos de software, 5*, págs. 69-70. Aguascalientes, México. Recuperado el 15 de 12 de 2017, de https://www.researchgate.net/profile/Claude_Laporte/publication/312874829_La_implementacion_de_la_norma_ISOIEC_29110_Guias_de_Gestion_e_Ingenieria_para_las_organizaciones_pequenas/links/5888aa00458515098e43a754/La-implementacion-de-la-norma-ISO-IEC-29110-

Loboguerrero, A. F., Castañeda Bueno, L., & Arboleda, H. F. (2011). Metodología ágil para equipos pequeños usando plataformas Microsoft. *S&T, 9*(18), 83-99.

López Zamarrón, U. d., Padilla Perea, C. D., Moran, G. E., & Berrelleza, E. (2017). *ITSON - Educar para Trascender*. Recuperado el 19 de 12 de 2017, de https://iswugaps2kanban.wordpress.com/ventajas-y-desventajas/

Maida, E. G., & Pacienzia, J. (2015). *Metodologías de desarrollo de software.* Tesis de Licenciatura en Sistemas y Computación, Pontificia universidad católica de Argentina Santa María de los Buenos Aires, Faculta de química e ingeniería, Buenos Aires, Argentina. Recuperado el 01 de 11 de 2017, de http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf

Melendez Valladarez, S. M., Gaitán, M. E., & Pérez Reyes, N. N. (2016). *Metodología ágil de desarrollo de software programación extrema.* Tesis, Universidad Nacional Autónoma de Nicaragua, Facultad de ciencias e ingeniería, Managua, Nicaragua. Recuperado el 15 de 12 de 2017, de http://repositorio.unan.edu.ni/1365/1/62161.pdf

Microsoft Developer Network. (2013). *MSDN*. Recuperado el 12 de 12 de 2017, de https://msdn.microsoft.com/es-es/library/dd997578%28v=vs.120%29.aspx?f=255&MSPPError=-2147217396

Ministerio de Tecnologías de la Información y Comunicación. (2015). *Informe de caracterización del sector de software y tecnologías de la información en Colombia.* Bogota, Colombia: Fedesoft.

Mittelman, J. H. (1996). *Globalization: Critical Reflections.* USA: Lynne Rienner Publishers.

Mogollón Afanador, J. O. (2010). *Adaptación de procesos ágiles de desarrollo de software al desarrollo individual de aplicativos pequeños y de bajo presupuesto.* Tesis de Maestría, Universidad de Pamplona, Pamplona, Colombia.

Mogollón Afanador, J., & Estaban Villamizar, L. A. (2011). El desarrollo individual de proyectos de software: Una realidad sin método. (U. d. Pamplona, Ed.) *Revista colombiana de tecnologías de avanzada, 1*(17), 51. Recuperado el 09 de 11 de 2017

Observatorio TI. (2017). *Observatorio TI.* Recuperado el 27 de 02 de 2019, de https://www.observatorioti.gov.co/

Organización de las Naciones Unidas para el Desarrollo Industrial (ONUDI). (2015). *Informe sobre el desarrollo industrial 2016. El rol de la tecnología y la innovación en el desarrollo industrial inclusivo y sostenible.* Viena: ONUDI.

Pazmiño Peña, D. A. (2013). *Metodología para la adaptación de agile en la implementación de una aplicación.* Trabajo de grado, Quito, Ecuador.

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2007). A Desing Science Research Methodology for Information System Research. *Journal of Management Informtion Systems, 24*(3), págs. 45-77.

Pressman, R. S. (2010). *Ingeniería del software un enfoque práctico* (Séptima edición ed.). México, México: McGraw-Hill.

Schwaber, K., & Shuterland, J. (2013). *La guía definitiva de Scrum: las reglas del juego.* ScrumGuides. Scrum.org and ScrumInc. Recuperado el 19 de 12 de 2017, de http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf

Sommerville, I. (2011). *Sofwtare engineering* (Novena edición ed.). Boston, Masachussets: Pearson Educación.

Soto Duran, D. E., & Reyes Gamboa, A. X. (2010). Introduciendo PSP (Proceso Personal de Software) en el Aula. *Revista Colombiana de Tecnologías de Avanzada, 2*(16), 1-5. Obtenido de http://www.unipamplona.edu.co/unipamplona/portalIG/home_40/recursos/03_v13_18/revista_16/27102011/01.pdf

Szulanski, G. (1996). Exploring Internal Stickiness: Impediments to the Transfer of Best Practice Within the Firm. *Strategic Management Journal, 17*, 27-43. Recuperado el 23 de septiembre de 2018, de http://www.jstor.org/stable/2486989

Trigas Gallego, M., & Domingo Troncho, A. (2012). *Metodología Scrum.* Universitat Oberta de Catalunya. Barcelona, España: Universitat Oberta de Catalunya. Recuperado el 19 de 12 de 2017, de http://hdl.handle.net/10609/17885

Turriza, A. (15 de diciembre de 2010). *Administrando proyectos de software*. Recuperado el 1 de septiembre de 2018, de Personal Software Process PSP: http://administrandoproyectos.blogspot.com/2010/12/psp.html

Watts, H. (2000). *The Personal Software Process (PSP) (CMU/SEI-2000-TR-022).* Carnie Mellon University: Software Engineering Institute. Recuperado el 31 de Agosto de 2018, de https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5283

WebFinance Inc. (2017). *BusinessDictionary*. Recuperado el 15 de 12 de 2017, de http://www.businessdictionary.com/definition/freelance.html

Wells, D. (1999). *The Rules of Extreme Programming*. Recuperado el 23 de septiembre de 2018, de http://www.extremeprogramming.org/rules.html

16 Anexo A. Encuesta para la prueba de concepto

PRUEBA DE CONCEPTO

Objetivo: determinar el nivel de comprensión del concepto del proceso de desarrollo unipersonal propuesto. (Para el logro del objetivo se empleará el sitio web diseñado en EPF Composer de Eclipse, que contiene todos los elementos del proceso de desarrollo unipersonal.)

Nombre: Correo electrónico:

Empresa: Formación:

1. ¿Cuál de los siguientes roles define mejor su actual puesto de trabajo?
    a. Administrador de proyectos
    b. Desarrollador/Programador
    c. Jefe de desarrollo
    d. Consultor
    e. Arquitecto de software
    f. QA/Pruebas/tester
    g. IT Staff/Soporte

2. ¿Cuál de las siguientes situaciones describe mejor su nivel de conocimiento en metodologías ágiles?
    a. Líder de equipo de desarrollo que usa metodologías ágiles
    b. Formo parte de un equipo de desarrollo que hace uso de metodologías ágiles
    c. Estoy considerando introducir metodologías ágiles en mi actual equipo
    d. Ha formado parte de equipos que hacen uso de metodologías ágiles
    e. He escuchado acerca de metodologías ágiles, conozco muy poco al respecto

3. ¿Por cuánto tiempo ha hecho uso personalmente de metodologías ágiles sin tener en cuenta desarrollos unipersonales o en equipos de desarrollo?
    a. Nunca
    b. Menos de 1 año

c. Entre 1 y 2 años
d. Entre 2 y 5 años
e. Más de 5 años

4. Desde su perspectiva y experiencia ¿cuáles considera usted pueden ser los beneficios del proceso de desarrollo unipersonal propuesto? (Marque con una x todos los que considere)
    a. Incrementa la productividad
    b. Mejora la calidad del software
    c. Mejora la productividad relación entre el desarrollador y el cliente
    d. Reduce los riesgos en el proceso de desarrollo unipersonal
    e. Mejora la disciplina del desarrollador unipersonal
    f. Mejora el proceso de mantenimiento y cambios de software

5. ¿El sitio web muestra claramente el proceso de desarrollo, describiendo claramente cada una de las etapas que este contempla?
    a. Si
    b. No
    c. Observaciones_____
    _____
    _____
    _____

6. ¿Considera que se conserva la agilidad en el proceso de desarrollo unipersonal propuesto formalizándolo mediante el esquema documental propuesto?
    a. Si
    b. No
    c. Observaciones_____
    _____
    _____

7. ¿El sitio web presentado es fácil de entender?
    a. Si
    b. No
    c. Observaciones_____
    _____
    _____

8. ¿El sitio web identifica de manera clara los roles de las personas que intervienen en el proceso de desarrollo unipersonal?
    a. Si
    b. No

9. ¿El diagrama del proceso es fácil de comprender y muestra la totalidad de este?
    a. Si
    b. No
    c. Observaciones_____
    _____
    _____

    Recomendaciones:
    _____