DEFINICIÓN DE UNA PRÁCTICA DE DESARROLLO DE SOFTWARE PARA EQUIPOS DE TRABAJO SIN RESTRICCIONES DE TIEMPO Y ESPACIO UTILIZANDO LOS PRINCIPIOS DE SEMAT

SANDRA MARCELA GUERRERO CALVACHE

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
DEPARTAMENTO DE SISTEMAS
PROGRAMA DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2018

DEFINICIÓN DE UNA PRÁCTICA DE DESARROLLO DE SOFTWARE PARA EQUIPOS DE TRABAJO SIN RESTRICCIONES DE TIEMPO Y ESPACIO UTILIZANDO LOS PRINCIPIOS DE SEMAT

SANDRA MARCELA GUERRERO CALVACHE

Trabajo de grado presentado como requisito parcial para optar al título de Ingeniero de sistemas

Director

Gonzalo José Hernández Garzón, Mg.

Co-Director:

Ing. Alexander Barón, Mg.

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
DEPARTAMENTO DE SISTEMAS
PROGRAMA DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2018

NOTA DE RESPONSABILIDAD

“Las ideas y conclusiones aportadas en este Trabajo de Grado son Responsabilidad de los autores.”

Artículo 1º del Acuerdo No. 324 de octubre 11 de 1966, emanado del honorable Concejo Directivo de la Universidad de Nariño.

“La Universidad de Nariño no se hace responsable de las opiniones o resultados obtenidos en el presente trabajo y para su publicación priman las normas sobre el derecho de autor”.

Artículo 13, Acuerdo N. 005 de 2010 emanado del Honorable Consejo Académico.

Nota de aceptación:

_____
_____
_____
_____
_____
_____
_____

_____

Firma Presidente del Jurado

_____

Firma del Jurado

_____

Firma del Jurado

San Juan de Pasto, Junio de 2018

AGRADECIMIENTOS

Expreso mi más sincero agradecimiento a Dios y a la Virgen María por darme la fortaleza y sabiduría para culminar exitosamente este trabajo de grado.

A mis padres por ser el motor que me acompañan día con día y me brindan su apoyo, cariño y comprensión.

A la Universidad de Nariño por permitirme ser parte de tan importante institución.

A la Vicerrectoría de Investigaciones, Postgrados y Relaciones Internacionales por brindarme su apoyo financiero a lo largo del proyecto.

A mi director de tesis el Ing. Gonzalo Hernández y mi codirector el Ing. Alexander Barón por el seguimiento brindado durante la investigación.

A todos los docentes del Departamento de Sistemas por brindar su conocimiento, entrega y profesionalismo en pro de formar estudiantes con alta disciplina y responsabilidad.

A mi amigo y colega Gerson Lázaro que sin lugar a duda estuvo allí brindándome sus consejos, su orientación y su tiempo a lo largo de mi investigación.

A los estudiantes Luis España, Daniel Tutistar, Juan Pablo Botina, Camilo Valencia por su disposición y buena voluntad en colaborarme en este proceso.

A mis amigos, los que con sus palabras de afecto siempre me animaron a alcanzar este objetivo.

DEDICATORIA

Este trabajo de grado se lo dedico primero a Dios porque él siempre ha estado conmigo en todo momento, en las buenas y en las malas.

A mi madre Mercedes Calvache y a mi padre Hernando Guerrero por ser las personas que siempre estuvieron para mí de forma incondicional y me dieron palabras de aliento para perseverar y alcanzar mis metas con compromiso y dedicación.

A mis abuelas Jesús Inés de Guerrero y Rosalba Santacruz por ser los motores que siempre me acompañaban.

A mis amigos que estuvieron para apoyarme en todo mi proceso y formación como Ingeniera de sistemas, ofreciéndome su ayuda incondicional y su cariño sincero y leal.

RESUMEN

La presente investigación muestra la formulación de una práctica de desarrollo de software para equipos que realizan sus actividades sin importar los contextos de tiempo y espacio, la cual es representada bajo el núcleo de SEMAT.

SEMAT es una comunidad de profesionales que hace un llamado a la acción en el campo de la Ingeniería de Software debido a los problemas relacionados con la aparición de diversos métodos sin una teoría universal que los sustente.

Es por eso que a través de este proyecto se pretende indagar en primer lugar las principales dificultades que tienen los equipos de desarrollo de software cuando estos trabajan de manera distribuida, identificar los elementos que harán parte la práctica tales como roles, actividades y herramientas, para luego adaptarla al núcleo de SEMAT en base a sus alfas de Requisitos y Sistema Software y que finalmente sea representada a través de su sintaxis gráfica.

PALABRAS CLAVE. SEMAT, alfas, espacios de actividad, competencias, equipo de desarrollo de software, tiempo, espacio, practica de desarrollo de software, métodos, núcleo, lenguaje, distribuido, desarrollo global de software, Ingeniera de Software.

ABSTRACT

The present work shows the formulation of a software development practice for teams that do their activities no matter time and space context. It is represented under the kernel of SEMAT.

SEMAT is a professional community that makes a call to action in the field of Software Engineering due to the associated problems with the apparition of different methods without universal theory that sustains them.

In this project I try to investigate the main difficulties that software development teams have when they work in a distributed manner, identify the elements that will be part of the practice, such as roles, activities and tools, in order to adapt them to the kernel of SEMAT based on its alphas of Requirements and Software System and it is finally represented through its graphic syntax.

KEYWORDS. SEMAT, alphas, activity spaces, competencies, software development team, time, space, software development practices, methods, kernel, language, distributed, global software development, Software Engineering.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INDICE DE ANEXOS

INTRODUCCIÓN

La importancia que ha tenido el software es contundente hoy en día, partiendo desde las necesidades que puede tener un individuo, hasta el control de procesos dentro de una organización, todo esto debido a que la sociedad requiere de herramientas que sirvan de apoyo a la toma de decisiones y a la realización de tareas que pueden ser un tanto complejas. Por ende, la industria del software abre sus puertas hacia cambios significativos y a explorar campos de innovación.

En la construcción de software es relevante tener en cuenta cada una de las fases de desarrollo que este concierne; donde la Ingeniería del Software trae a su paso metodologías o modelos para ser implementadas dentro de los equipos, lo que además contribuye a la definición de pautas y criterios que busquen la obtención de un producto de calidad y propicien el éxito del proyecto.

Existen muchos enfoques para la creación de software desde modelos prescriptivos hasta procesos basados en metodologías de agilísimo. Todos estos concentrados en el propósito de cumplir con ciertos objetivos.

Actualmente algunos equipos de desarrollo emplean métodos convencionales, otros escogen métodos que les proporcionen flexibilidad al momento de entregar un esfuerzo de software, o incluso lo seleccionan según las tendencias del momento.

Sin embargo, la existencia de tantos modelos ha generado que las diversas organizaciones que se enfocan en esta área escojan alguno de los tantos de la lista para la ejecución de dichos procesos, dejando de lado la necesidad de poseer una teoría única que sustente a la Ingeniería de Software, tal y como lo propone el núcleo SEMAT (Software Engineering Method and Theory).

SEMAT es una organización formada por profesionales que ha traído la perspectiva de consolidar un núcleo común de elementos que contribuyan a formar un lenguaje simple y entendible para todos los que trabajan con la Ingeniería de software generando medidas específicas que no dependan de un método particular sino por el contrario se traslade al uso de prácticas que fomenten el crecimiento de mejora para el equipo de desarrollo y la estructuración de ideas globales que sirvan como sustento teórico científico. Esta comunidad representada a través de su estándar Essence destaca cuatro elementos importantes en la Esencia de la Ingeniería de software: los métodos, las prácticas, el núcleo y el lenguaje.
La presente investigación bajo este lineamiento, pretendió definir una práctica que abarcó el desarrollo de software teniendo en cuenta las circunstancias que

manejan algunos equipos cuando estos trabajan bajo diferentes contextos de tiempo y espacio, agregándole la participación de cada uno de los miembros, el estímulo y la motivación requerida para la realización de sus actividades para finalmente ser representada bajo el núcleo de SEMAT.

El documento está estructurado de la siguiente manera. En primer lugar, se detallaron las dificultades, ventajas y desventajas que presentan los desarrolladores cuando estos trabajan de manera remota obtenidas tras la lectura de información bibliográfica y a través un acercamiento efectuado a equipos de desarrollo de software de la ciudad de Pasto. En la siguiente sección se identificaron modelos aplicados en algunas organizaciones cuando trabajan bajo este tipo de circunstancias. En base a dicho análisis se continuó con la formulación de la práctica relacionando los roles, actividades y herramientas de apoyo que harán parte de ella para posteriormente enlazarla a los principios de SEMAT y representarla gráficamente de acuerdo a los elementos que el estándar involucra.

PRESENTACIÓN DEL PROYECTO

TÍTULO

Definición de una práctica de desarrollo de software para equipos de trabajo sin restricciones de tiempo y espacio utilizando los principios de SEMAT.

GRUPO Y LÍNEA DE INVESTIGACIÓN

La modalidad del proyecto corresponde a un trabajo de Investigación, que se desarrolló bajo la línea de ingeniería de software y manejo de Información.

La línea de Ingeniería de Software y Manejo de Información tiene como objetivo abarcar todas las etapas del desarrollo de software soportados bajo unos constructos teóricos, así como también de planificar, analizar, diseñar, implantar y administrar sistemas complejos de información y de conocimiento asegurándolos con un compromiso de calidad.

DELIMITACIÓN DE LA INVESTIGACIÓN

La presente investigación tuvo el propósito de definir una práctica de software para equipos sin restricciones de tiempo y espacio que permita la organización de actividades durante la fase de desarrollo o codificación.

PLANTEAMIENTO DEL PROBLEMA

La Ingeniería de Software es una disciplina indispensable en el desarrollo de software puesto que trae consigo fuertes fundamentos teóricos que permiten establecer cuan importantes son para crear proyectos de pequeña y gran escala basados en criterios de conocimiento e investigación tratando siempre de brindar la eficiencia y la calidad en todos sus procesos.

Actualmente, existen muchos métodos de desarrollo de software, desde los tradicionales hasta los relacionados con el enfoque ágil que han traído diversas perspectivas de crear software en equipos arraigándose a un modelo de trabajo en específico y con una terminología propia de este. Lo anterior ha causado que la Ingeniería de Software se convierta en un sin fin de conceptos, teniendo claro que incluso muchos de ellos significan lo mismo de un método a otro, pero como han adoptado sus propios términos, la universalidad de estos se ha ido perdiendo.

Entre las diversas modalidades para construir software, existen algunas en donde las actividades se llevan a cabo en diferentes contextos de tiempo y espacio. Sin embargo, muy pocos modelos definen la forma de trabajo adecuada para este enfoque de desarrollo. Hoy en día, las organizaciones ya han optado por trabajar con personas de manera remota, logrando así que exista un intercambio cultural, colaborativo y de continuo aprendizaje, gracias a las capacidades de los

integrantes que no necesariamente están arraigados a un lugar establecido, sino que por el contrario pueden encontrarse desde cualquier parte del mundo para brindar ese conocimiento y participar de manera activa en un proyecto de este tipo.

Con lo anterior se puede inferir que debido al impacto que ha generado esta modalidad de trabajo remoto con sus posibles ventajas como la flexibilidad de horarios, la realización de tareas de manera independiente, la reducción de costos en transporte y demás; y pese a las dificultades en que los miembros pueden verse afectados, tales como el desfase de horas, diferencias en el idioma, transición de relaciones interpersonales por virtuales, dificultad en la toma de decisiones entre otros, nace la posibilidad de romper dichas barreras y convertirlas en oportunidades que permitan proponer una práctica que aplique a estas circunstancias, y que consecuentemente cualquier desarrollador a futuro pueda implementarla en cada esfuerzo de software que realice de manera cooperativa permitiéndole además la organización de sus actividades.

SEMAT aparece para darle un nuevo sentido a la Ingeniería del software. Bajo sus dos principios fundamentales busca encontrar un núcleo común de nociones aceptadas, así como también consolidar la definición de una teoría sólida que permita evaluar el progreso del desarrollo de software, las prácticas usadas, implementar nuevas ideas para mejorar la comunicación entre el equipo y por ende su metodología de trabajo adoptando ante todo medidas globales.

Mediante dicho núcleo y con la apropiación de sus elementos se pretendió establecer una práctica de software enfocada en grupos de trabajo donde la creación de aplicativos se basa en hacerlo a distancia y con aportes colaborativos de diferentes personas sin importar el lugar donde lo realicen, zona horaria, ni el tiempo que empleen para cumplir dicho objetivo arraigado así a sus propósitos fundamentales y la contribución de conocimiento en esta área.

FORMULACIÓN DEL PROBLEMA

¿Cómo definir una práctica de desarrollo de software que permita abarcar contextos para equipos de trabajo que se encuentran en diferentes lugares y tiempos, alineados hacia un compromiso colaborativo y enlazados con los principios de SEMAT?

SISTEMATIZACIÓN DEL PROBLEMA

- ¿Cómo la práctica de desarrollo de software enfocada a marcos de trabajo de espacio y tiempo puede contribuir con los principios propuestos por SEMAT?
- ¿Qué alfas del Núcleo SEMAT abarcará esta práctica de desarrollo?

- ¿Cómo puede esta práctica ayudar a los equipos de desarrollo en la realización de sus actividades cuando estos trabajan en diferentes contextos de tiempo y espacio?

JUSTIFICACIÓN

SEMAT es una comunidad de investigadores de presencia mundial que ha generado el llamado a la acción en el campo de Ingeniería de Software para lograr el intercambio de ideas y puntos de vistas que permitan la consolidación de un constructo teórico que soporte y mejore los procesos en esta área. Esta surgió debido a los problemas que la Ingeniería de Software presenta hoy en día destacando la ausencia de una teoría conceptual que la sustente científicamente y que contribuya a formular los elementos claves para complementar las prácticas de desarrollo y fortalecer aquellas que ya están en uso.

Además, diversas organizaciones implementan muchas metodologías en la ejecución de sus proyectos, cambiando de una a otra cuando presentan dificultades o haciendo un hibrido entre ellas. SEMAT plantea cambiar ese paradigma de uno basado en métodos a uno enfocado en prácticas, para que estas se integren a un formato común en donde toda la comunidad las entienda y pueda aportar a las mismas y las apropie a sus formas de trabajo.

Con ello el desarrollo de software requiere de una atención especial en cada una de sus etapas, más aún cuando se trabajan en marcos de colaboración. Actualmente el conjunto de metodologías existentes, plantean su enfoque para equipos de trabajo que efectúan sus actividades de manera presencial siendo estas utilizadas en diferentes entornos. Sin embargo, en algunas circunstancias este factor es muy difícil de concretarse, debido a que existen grupos de trabajo que consolidan sus proyectos de software desde diferentes lugares del mundo y a través de internet. Esta situación es un tanto especial debido a que el contexto en que lo realizan difiere mucho de la localización en donde el desarrollador se encuentre y el momento en que este puedan realizar su aporte colaborativo.

A raíz de la situación presentada y en relación a los objetivos de SEMAT se buscó definir una práctica de desarrollo donde se involucre a equipos que trabajan bajo estas condiciones, teniendo en cuenta cada uno de los parámetros que influyen dentro de este aspecto y abarcando los elementos del núcleo propuestos.

La investigación contribuirá a la mejor organización de trabajo colaborativo entre diversas personas estableciendo una práctica sólida que tenga entradas y salidas satisfactorias con un impacto de carácter social y de gran aprendizaje en donde las barreras de tiempo y espacio sean una dificultad para lograr el objetivo deseado.

OBJETIVOS

Objetivo General

Definir una práctica de desarrollo de software para equipos que trabajan en diferentes contextos de espacio y tiempo utilizando los principios de SEMAT.

Objetivos Específicos

- Determinar las dificultades que presentan los equipos de desarrollo de software cuando estos trabajan en diferentes contextos de tiempo y espacio.
- Recopilar y analizar información sobre metodologías o modalidades de trabajo existentes que se relacionen con el desarrollo de software de manera distribuida.
- Establecer los roles, actividades y herramientas a involucrar dentro de la práctica de desarrollo de software aplicable a estos contextos.
- Relacionar el estudio realizado en base a los principios de SEMAT.

1. MARCO TEORICO

Es importante destacar que debido a que la práctica de desarrollo enfocada a contextos de espacio y tiempo no está claramente definida, se decide abarcar en primer lugar algunos de los modelos de desarrollo presentes dentro de la Ingeniería de software que se relacionen con marcos de cooperación y que permitan establecer el soporte conceptual para la investigación.

1.1 FUNDAMENTOS TEORICOS

1.1.1 Los Modelos de Desarrollo de Software

Tradicionales

- **Modelo de cascada**: Es un modelo perteneciente a los modelos tradicionales de la ingeniería de software el cual se caracteriza por que la forma en cómo se desarrolla el software es de manera secuencial- lineal, es decir que si no termina una etapa no puede seguir con la siguiente y así sucesivamente, motivo de desventaja para el desarrollador pues es complicado seguir tan rígidamente estos procesos. [1] [2]
  Análisis del Modelo. Del método cabe resaltar que este se concentra en realizar todo linealmente siguiendo una estructura muy rígida y poco flexible y no tiene tan presente el papel de los miembros del equipo. Fue uno de los principales pioneros que contribuyeron como base a nuevas metodologías de desarrollo de software.

- **Modelo proceso incremental**: Es un modelo mucho más estratégico puesto que maneja las entregas del software como incrementos y estos cada vez contienen detalles más perfeccionados de la entrega anterior y más acorde a los requerimientos expuestos por el cliente. Cabe destacar que con tan solo la primera entrega ya se cuenta con gran parte del software solicitado y los siguientes avances ya son mejoras mucho más significativas. Adicionalmente el ingreso de más personas al proyecto para su desarrollo en cualquier etapa no afecta en absoluto el rendimiento del mismo. [1] [2]
  Análisis del Modelo. Este modelo se destaca la integración de más participantes de una manera independiente en cualquier fase del proceso, es mucho más abierto a esa posibilidad y evita caer en la realización de trabajo repetitivo.

- **Modelo de proceso evolutivo**: Son modelos orientados a desarrollar versiones de software de manera iterativa permitiendo una retroalimentación de cada etapa. Entre estos se destacan: [1] [2]

Enfocado en la realización de prototipos: como su nombre lo indica, la idea de este modelo es realizar prototipos de software y emplear en las etapas los más adecuados de acuerdo a los requisitos que fueron planteados. Comienza por una planeación de los mismos y la realización de un diseño inicial con las características más relevantes de ellos para seguir con la etapa de desarrollo hasta llegar al final tomando como referencia esos prototipos que bien pueden hacer parte directa del producto software o bien solo son desechables en el sentido en que solo fueron empleados para ciertos momentos y tras la evolución presentada no fue necesario más su uso. [1] [2]

Análisis del modelo: Este modelo trata de brindar una ayuda al desarrollador cuando este presenta algún tipo de inseguridad sobre el algoritmo que debe emplear y también cuando el cliente presenta los objetivos que va requerir el software de una manera general sin detallar a gran profundidad cada uno de estos.

Modelo Espiral: Es un modelo que involucra más profundamente las etapas de construcción de software desde la planeación, seguida por el modelado, la construcción, el despliegue y la comunicación de manera cíclica, con entregas evolutivas y tomando en cuenta adicionalmente un aspecto muy importante: El riesgo en cada uno de los procesos [2] [3]

Análisis del modelo: Este modelo se destaca de los anteriores puesto que estima el riesgo que puede existir a lo largo de las etapas de desarrollo de software, tomándolo con un factor importante dentro de ello. Adicionalmente al hacer esto, formula estrategias para disminuirlo verificando que cada una de las fases se haga de la mejor manera posible y planificando los recursos necesarios para el cumplimiento de las metas del proyecto.

- **Modelo concurrente:** Es un modelo que maneja un control mucho más centralizado en cada uno de los procesos de desarrollo de software, permitiéndole obtener una perspectiva de la situación actual del proyecto y los eventos que de alguna u otra manera están inactivos o bien están en curso. [2] [4]

  Análisis del modelo: Es un modelo el cual permite proporcionar el estado actual del proyecto en donde las actividades que se catalogan en el mismo se pueden llevar simultáneas con otras lo que permite ver y analizar las transiciones que existen y que todas las personas del equipo actúen al mismo tiempo.

Modelos de proceso especializado

- **Modelo de proceso de personal de software (PSP)**: Es un modelo que involucra la calidad dentro del desarrollo del proyecto. Abarca cinco etapas importantes: [5]
  - ✓ **Planeación**: Hace un estimado general de los recursos y de los costos que va a tener el producto software a desarrollar.

- ✓ **Diseño de Alto Nivel**: Exalta ante todo la importancia de cada componente necesario dentro de los procesos de software elaborando un modelo adecuado de los mismos.
- ✓ **Revisión del diseño de alto nivel**: Consta en verificar si el diseño realizado es el correcto y detecta a tiempo cualquier anomalía presentada.
- ✓ **Desarrollo**: Mejora y revisa el diseño del componente consolidándolo ya en un desarrollo mucho más robusto teniendo en cuenta las pruebas pertinentes.
- ✓ **Post Mortem**: Mide la eficacia y eficiencia de las actividades realizadas.

Análisis del modelo: Este modelo destaca la manera en cómo una persona distribuye su trabajo y tiempo en el desarrollo de un proyecto software, mediante la realización de prácticas adecuadas con la pertinente disciplina y siempre con calidad. Destaca ante todo la organización de las actividades de manera individual que pueden ser clave para la estructuración de un trabajo grupal tal y como lo hace TSP.

- **Modelo del proceso de equipo de software (PES o TSP):** A diferencia de PSP, PES trabaja la calidad en el marco del trabajo colectivo, consolidando equipos fuertes con alto sentido de responsabilidad, pero sobre todo que sepan planificar buenas estrategias para llevar a cabo sus tareas de la mejor manera posible y basados en estándares de madurez CMM. Las actividades que involucra son desde el inicio del proyecto, el diseño de alto nivel, la implementación, la integración y pruebas y finalmente el post mortem.
  Cabe resaltar que este tipo de modelos los miembros de un grupo trabajan por un objetivo en común y son excelentes en liderar de la mejor manera posible los niveles de cada proyecto que se les proponga. “TSP busca integrar un equipo que tenga como punto de partida la unificación del mismo, para poder llevar a cabo todos aquellos procedimientos que puedan realizar mejora a los procesos que desarrollan.” [5]
  Análisis del Modelo: De este modelo se traduce más específicamente la idea de trabajar en equipo como un aspecto vital en el desarrollo de software, el cual contribuye al aporte de ideas por parte de cada uno de los integrantes además que maneja estándares propios de Ingeniería contribuyendo a la correcta planificación de los proyectos y estableciendo metas claras, claves para tener un buen rendimiento. Se destaca también la parte de motivación en donde cada miembro hace un aporte significativo al proyecto siempre en miras de lograr el éxito de este y lograr una comunicación pertinente con el resto del equipo.
  Al estar fuertemente consolidados detectan errores a etapas tempranas del desarrollo y proveen la manera de solucionarlos de la mejor manera posible sin que genere afectaciones a futuro.

Enfocados en el agilismo.

- **Metodología Ágil de Programación Extrema (XP):** Es uno de los enfoques más empleados dentro de la metodología de agilísimo caracterizado por llevar cada una de las etapas con mucho cuidado.
  Comenzando por la planeación en donde se maneja un conjunto de historias (así denominadas por el método) en donde el cliente expone sus necesidades, lo que espera a futuro del software y las organiza de acuerdo a un nivel de prioridad. Luego ya por parte del equipo, se encarga de observar cada historia y darle un estimado de costo y de tiempo y si esté supera las 3 semanas, se pide que se descompongan dichas historias en segmentos mucho más pequeños y se repite el proceso.
  Como segunda etapa sigue el diseño, destacando un modelo comprensible y apto de disminuir el riesgo en todo sentido. Siguiente a ello está la codificación implementadas de acuerdo al orden de las historias propuestas inicialmente y realizando en cada de ellas las pruebas unitarias pertinentes y el desarrollo se efectúa por parejas consolidando y socializando cada avance a todo el equipo cada día para detectar errores a tiempo. Finalmente se culmina con la etapa de pruebas manejando de tres tipos: las unitarias previas a la codificación, las de integración y validación del lado de detección de anomalías y las de aceptación frente a si se cumplió con las expectativas del cliente. [6] [7]
  Análisis del Modelo: Es uno de los modelos más empleados dentro del enfoque ágil destacando la comunicación dentro del equipo, el fomento a las buenas relaciones interpersonales, el aprendizaje para cada uno de los miembros y la organización y distribución de tareas especialmente en el desarrollo donde este se hace en binas, haciendo de cada actividad algo agradable con un clima de trabajo amigable.
  Además, se rige bajo 5 valores (comunicación, simplicidad, retroalimentación, valentía y respeto) junto con doce prácticas de desarrollo las cuales son la guía que deben seguir y que contribuyen a lograr un producto software apto para enfrentar cambios.

- **Metodología de Desarrollo Adaptativo de software (DAS):** Esta metodología se basa principalmente en incluir dentro de sus principios la organización y colaboración del equipo de trabajo basándose específicamente en una buena planeación del proyecto a desarrollar, la comunicación eficaz entre los miembros del grupo fundamentada con valores de tolerancia, respeto y responsabilidad sobre los aportes e ideas que cada uno realiza, siempre manejando las críticas de una manera positiva y en miras de mejorar. A partir de esto lograr el aprendizaje en cada una de las fases, con las revisiones continuas y pertinentes para lograr el éxito del producto.
  Esta metodología maneja más acertadamente el compromiso del trabajo en grupo resaltando principios por el cual el equipo debe seguir y que contribuyan como una guía a cumplir y un proceso de aprendizaje para todos los miembros. [8]

Análisis del Modelo: Este modelo también destaca valores importantes dentro de las relaciones entre los miembros del equipo estableciendo el respeto como un principio fuerte y clave para lograr la cooperación tanto entre desarrolladores y clientes, lo que conllevaría a establecer un proyecto acorde a unos requerimientos que bien estos pueden ser adaptable a cambios. Maneja tres criterios importantes la especulación, la colaboración y el aprendizaje.

- **Metodología Scrum:** Scrum es un modelo muy empleado hoy en día debido a la efectividad en la realización de cada una de las etapas de desarrollo de software. Esta se basa en entregas parciales del producto final manteniendo una relación mucho más estrecha con el cliente puesto que se ajusta a lo largo del proyecto a posibles cambios en los requerimientos presentados inicialmente.
  Su principal objetivo es cumplir con las necesidades del cliente empleando el menor número de recursos (tiempo y dinero) logrando así la optimización de los mismos. Maneja diferentes roles entre los cuales se destaca el de Product Owner y el Scrum Master sin dejar de lado al desarrollador, el tester y por supuesto el cliente.
  Para la revisión de los trabajos asignados al equipo el cual se compone por un conjunto de historias de usuario (Product Backlog), Scrum maneja reuniones diarias de 15 minutos para determinar el avance de las actividades y exponer las posibles inconsistencias que se pudieron haber presentado durante su realización. [8]
  Análisis del modelo: La filosofía de Scrum es realmente valiosa puesto que pretende lograr la productividad de los equipos de desarrollo logrando la entrega de software funcional a corto plazo. Las reuniones diarias que practica este modelo permiten estar al tanto de los avances del equipo y detectar a tiempo posibles errores. Aplica características de adaptabilidad, productividad y flexibilidad.

- **Método RUP:** Es un método adoptado por las organizaciones para lograr la productividad del equipo en el desarrollo de un producto software. Su filosofía basada en la metodología orientada a objetos y en el uso de UML. Sus prácticas son soportadas y probadas puesto que adopta herramientas propuestas por IBM. Sus principales características se rigen por la implementación de diagramas de casos de uso resaltando cada uno de los requerimientos expuestos por un cliente; además es un modelo centrado en la arquitectura es decir asimila el proyecto desde diferentes perspectivas y visiones lo que garantiza la mejor comprensión del sistema a desarrollar y finalmente que es un modelo iterativo incremental en donde cada iteración consolida una parte funcional del producto final. [9]
  Análisis del método: Al ser un método bastante robusto requiere de un buen número de programadores para trabajar más acordemente, adicional que entre todos los integrantes manejan en común una base de conocimiento, el

proceso, la manera de cómo desarrollar el software y el lenguaje de modelado UML. Trabaja sus cuatro fases (inicio, elaboración, construcción y transición) con altos criterios de calidad y ofrece ventajas sobre otros métodos.

- **Modelo de desarrollo de software por componentes.** Se trata de un modelo muy comúnmente empleado para entornos abiertos cuyo objetivo principal es la reutilización de componentes en diferentes contextos para los que haya sido creado permitiendo así la creación de sistemas más potentes y verdaderamente robustos**.** Tal así que se define a un componente como una unidad de aplicaciones software que parte de una serie de requerimientos, involucra un diseño y que está listo para ser desarrollado e implementado en un sistema al lado de otros componentes. [10]
  Este modelo además de separar al sistema por componentes para su mejor análisis y desarrollo busca que estos sean reutilizados en otros entornos y que además se propicie el intercambio de información entre ellos de la mejor manera posible.

1.1.2 SEMAT (Software Engineering Method and Theory)

SEMAT fue creada tras la iniciativa de Ivar Jacobson en el año 2009. Es una comunidad de alrededor 2000 personas que ha ganado reconocimiento de talla internacional la cual cuenta con el apoyo de diversas empresas y organizaciones, centros de educación superior y profesionales, justamente para trabajar en conjunto por consolidar una teoría que soporte la Ingeniería de Software y principios hacia la realización de mejores prácticas. [11]

A raíz de las investigaciones efectuadas en este campo, SEMAT cuenta con la presencia de siete capítulos los cuales se encuentran en diferentes lugares del mundo destacando China, India, Japón, Corea, América Latina, Rusia y Sudáfrica teniendo representantes de todos los gremios.

El capítulo latinoamericano se encuentra en Colombia específicamente en la Universidad Nacional de Colombia sede Medellín cuyo director es el Dr. Carlos Mario Zapata Jaramillo.

SEMAT tiene dos principios fundamentales los cuales los expresa a través de su estándar Essence y son:

- Encontrar un núcleo común de elementos aceptados.
- Definir una teoría sólida que soporte a la Ingeniería de Software.

Adicionalmente se destaca que el núcleo de SEMAT posee las características de ser accionable, extensible y práctico donde además se enlaza con siete alfas: oportunidad, interesados, requisitos, sistema de software, trabajo, forma de trabajo y equipo dándole un sentido mayor de estudio a estos parámetros.

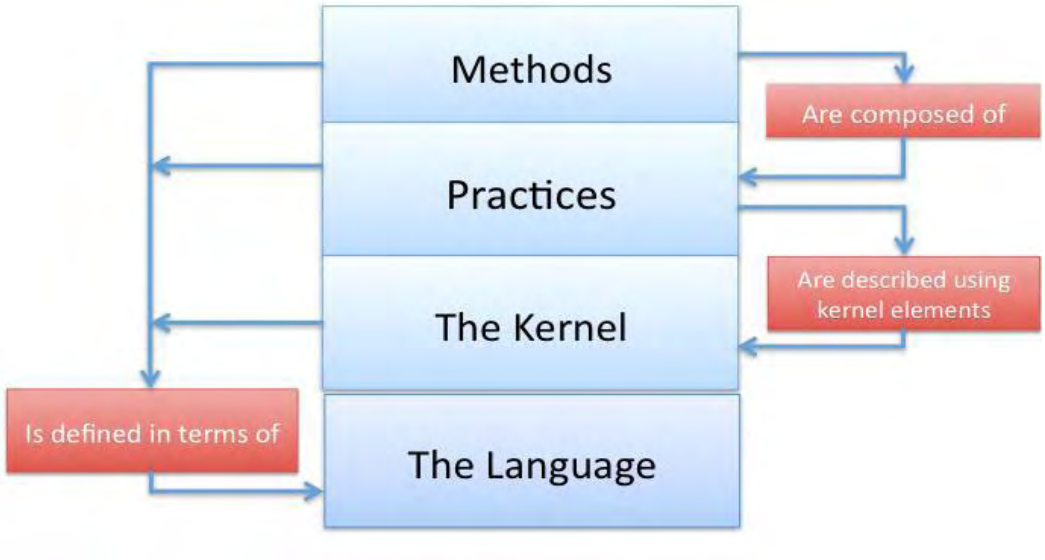
“La esencia es una de las primeras salidas más prominentes de la comunidad SEMAT generando un estándar para trabajar con métodos en la Ingeniera de software adoptado por OMG en junio de 2014.” [12]

Conceptualización de SEMAT

En [13] se habla que SEMAT a través de Essence consolida un marco de pensamiento que involucra restructurar la Ingeniería de Software y mejorar los métodos que se relacionan. Es por eso que para la conceptualización del estándar es necesario reconocer una serie de aspectos que son de vital importancia y que permiten ampliar de manera clara el objetivo para el cual fue creado.

- **Métodos.** El estándar define a los métodos como un conjunto de prácticas que pueden ser usadas y que además representan las actividades que se llevan a cabo durante el desarrollo.
- **Práctica.** Se define como la forma de abordar una actividad y que puede ser repetitiva cumpliendo un propósito en específico. Además, una práctica puede ser reutilizable por cualquier método.
- **Núcleo**. Contiene los elementos claves de la ingeniería de software y hace parte de los métodos.
- **Lenguaje**. Es la forma de definir los métodos, las prácticas y el núcleo.

Figura 1. Elementos de Essence



Fuente. [13]

El núcleo de Essence.

Este núcleo lo que pretende es proporcionar un conjunto de elementos que permitan establecer de mejor manera las prácticas de desarrollo de software y definirlo de la mejor manera. Dichos elementos son las alfas, los espacios de actividad y las competencias las cuales a su vez precisan tres áreas relevantes: la parte del cliente, la de solución y la de esfuerzo. [13]

Alfas del núcleo

En [14] [13] "Las alfas se pueden definir como la representación de las "cosas esenciales para trabajar". Dichas alfas dimensionan el esfuerzo de ingeniería de software que un equipo realiza y están constituidas por un conjunto de estados, donde cada uno de estos contiene una lista de chequeo que permite ver el progreso que va teniendo.

De acuerdo a las áreas mencionadas, las alfas que se encuentra en la parte del cliente son:

- **Alfa Oportunidad**: "Es el conjunto de circunstancias adecuado para desarrollar o modificar un sistema de software. La oportunidad articula la razón para la creación de lo nuevo o modificado del sistema de software" [14] [13].
  En esta alfa se destaca la posibilidad de contribuir a la creación o mejora de ciertas herramientas de software con el criterio de solventar una necesidad y buscando innovar y generar propuestas acordes a las tendencias del momento.

- **Alfa Interesados**: "Las personas, grupos u organizaciones que afectan o se afectan con un sistema de software. Los interesados proporcionan la oportunidad y son la fuente de los requisitos y la financiación para el software sistema" [14] [13]. El alfa interesados involucra a todos los agentes tanto externos como internos que de una u otra manera intervienen directa o indirectamente en un proyecto de software trayendo consigo aportes relevantes que contribuyen a la consolidación de los elementos y características a definir dentro del mismo y exponiendo un conjunto de necesidades que buscan ser resueltas.

En el área de solución están:

- **Alfa Requisitos**: "Lo que el sistema de software debe hacer para tratar la oportunidad y satisfacer a los interesados" [14] [13]. Esta alfa parte de una serie de necesidades expuestas por un cliente las cuales son el punto de partida al momento de desarrollar el software ya que serán los parámetros de entrada a construir.

- **Alfa Sistema de software**: "Un sistema se compone de software, hardware y los datos que proporciona el valor primario de la ejecución del software". [14] [13]. Esta alfa agrupa todos los elementos tanto físicos como lógicos de un sistema destacando la importancia de que juegan cada uno de estos.

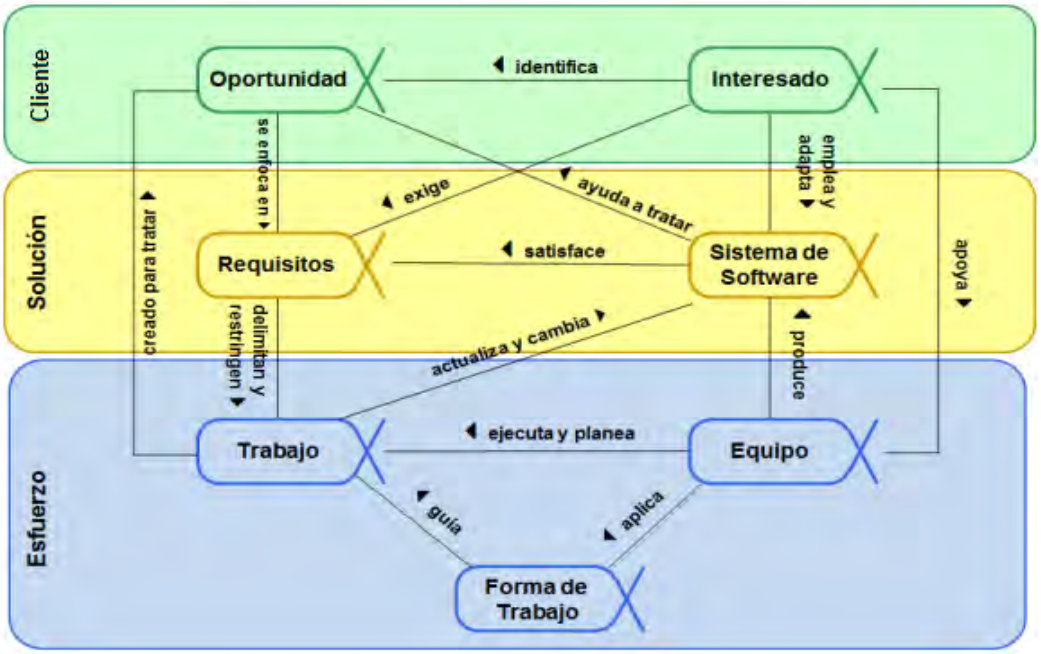
Y finalmente en el área de esfuerzo se encuentran:

- **Alfa Trabajo**: "Actividad que implica un esfuerzo mental o físico realizado con el fin de lograr un resultado. En el contexto de la ingeniería de software, el trabajo es todo lo que hace el equipo para cumplir con las metas de producción de un sistema de software que coincida con los requisitos" [14] [13]. El alfa de trabajo es muy relevante considerarla puesto que abarca cada una de las actividades que el equipo de desarrollo realiza para lograr con sus objetivos y el esfuerzo que invierten en ello.

- **Alfa Equipo:** "Un grupo de personas que participa activamente en el desarrollo, mantenimiento o entrega de un sistema de software" [14] [13]. El alfa de equipo sin lugar a duda define un conjunto de personas con la capacidad y el criterio de efectuar una tarea con responsabilidad y aportar de manera idónea al desarrollo del proyecto de software.

- **Alfa Forma de trabajo:** "El conjunto adaptado de prácticas y herramientas que utiliza un equipo de orientar y apoyar su trabajo" [14] [13]. La forma de trabajo reúne todo tipo de prácticas que contribuyen a que el equipo las implemente de manera eficiente y se efectúe su labor de la mejor manera posible y organizada.

Figura 2. Alfas del Núcleo de SEMAT.

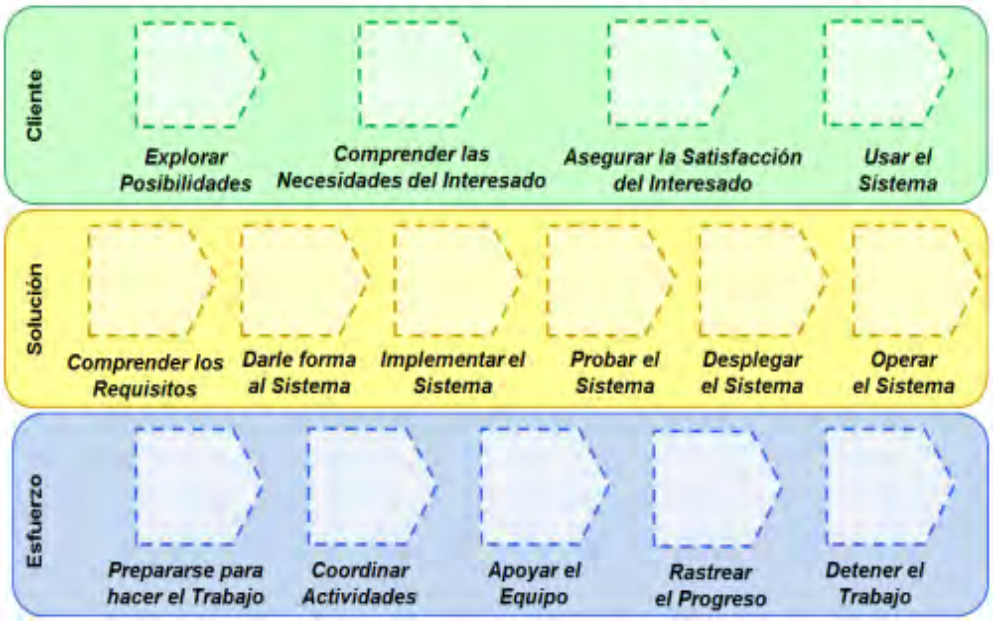


Fuente. Extraído de [11]

Espacios de actividad.

Los espacios de actividad se definen como las cosas que siempre hacemos y complementan cada una de las alfas. Se agrupan también de acuerdo a las áreas de cliente, solución y esfuerzo tal y como se muestran en la Figura 3.

Figura 3. Espacios de Actividad



Fuente. Extraído de [11]

Competencias.

Resaltan las capacidades o habilidades que se tienen para realizar una actividad que implique trabajo. Por el área de cliente se encuentra la representación del interesado, en la parte de solución está el análisis, el desarrollo y las pruebas y finalmente en la parte de esfuerzo está el liderazgo y gestión.

Figura 4. Competencias del núcleo

Fuente. Extraído de [14] [13]

1.2 ANTECEDENTES

1.2.1 BASADOS EN MODELOS DE TRABAJO

- **Prácticas, metodologías y herramientas para la gestión y ejecución de proyectos de desarrollo de software que utilicen equipos de trabajo geográficamente distribuidos**

  Autor: Francisco Javier Ramírez Morales

  Modalidad: Tesis de trabajo de grado para Magister en Ciencias de la Ingeniería

  Año: 2008

  Análisis del estudio

  Este trabajo de grado de maestría pretende hacer un exhaustivo análisis sobre el desarrollo distribuido de software principalmente resaltado algunos factores que puedan afectar este tipo de modalidad. Además, comparte algunas prácticas de desarrollo de software de manera presencial que son realmente importantes y contribuyen al trabajo colaborativo y conjunto, para luego

profundizar en algunas que fueron aplicables por organizaciones que en algún momento tuvieron sus equipos de manera deslocalizada. El aporte de la tesis es muy relevante para la presente investigación puesto que es soporte vital sobre algunas formas de trabajo ya utilizadas o sugeridas y lógicamente de los problemas presentados en este tipo de circunstancias. [15]

- **Remoto**

  Autor: Jason Fried

  Modalidad: Libro **Año**: 2014

  Análisis del estudio

  Este documento detalla más el trabajo a distancia como una modalidad nueva y utilizada ahora por las organizaciones para propiciar más productividad, adquirir talento diversificado y mostrar las ventajas de salir de la oficina y laborar desde casa. Además, comparte el autor la experiencia de una empresa creada por el mismo llamada 37signals el cual sigue los principios del trabajo a distancia y consolida la libertad de colaborar sin tener un espacio físico definido abriendo la posibilidad de instaurar el teletrabajo. Este libro comparte sugerencias para llevar a cabo de manera remota una actividad y especialmente para líderes empresariales que ponen en marcha sus negocios, poniendo a prueba que tanto jefes y empleados pueden interactuar sin barreras de lugar o tiempo. Las ideas que proponen se fundamentan principalmente en “Llevar el trabajo a los trabajadores y no los trabajadores al trabajo”. [16]

- **Metodología para el desarrollo colaborativo de software libre**

  Autores: Johanna Álvarez, Víctor Bravo. - Fundación Centro Nacional de Desarrollo e Investigación en Tecnologías Libres CENDITEL.

  Versión: 2

  Año: 2015

  Análisis del estudio:

  La metodología propuesta con anterioridad resalta prácticas cooperativas en un equipo de trabajo como un factor vital que estimula la colaboración y el cumplimiento de los objetivos, así como también la adquisición de conocimiento que contribuya al crecimiento del proyecto a efectuar. Todas las experiencias que puedan generarse a raíz de esto son bastante enriquecedoras puesto que se acoplan a los criterios de consigo una serie de recomendaciones en base a los resultados obtenidos tras la práctica realizada

que sirven de apoyo no solo para desarrolladores sino en el nexo de comunicación con el cliente e interesados. [17]

- **Ambiente distribuido aplicado a la formación/capacitación de RR HH: un modelo de aprendizaje cooperativo-colaborativo**

  Autor: Ing. Fernando Javier Lage

  Año: 2001

  Análisis del Estudio: [18]

  La tesis realizada hace varios años trato de un enfoque colaborativo-cooperativo basándose en un método tradicional implementando varias herramientas para lograr la efectividad del modelo.

  Resalta el trabajo en equipo como un elemento donde deben existir una serie de factores básicos destacando la confianza para desenvolverse en el equipo, la buena práctica hacia la comunicación, el apoyo y colaboración entre compañeros, evitar confrontarse unos a otros con argumentos inválidos y solo por tener puntos de vista diferentes se debe manejar este tipo de circunstancias con el respeto pertinente, poseer la habilidad para trabajar grupalmente y manejar el liderazgo como una oportunidad de dirigir con criterio a unas personas y establecer claramente los roles de cada uno con alto sentido de pertenencia y responsabilidad.

  Como parte de los ambientes de aprendizaje colaborativo resaltan a formar equipos dentro de este tipo de entornos en donde los integrantes participan activamente en cada una de las etapas, en los aspectos de interés del proyecto y los problemas a resolver.

  Si se habla de modelo cooperativo por otro tanto resaltan la asignación de tareas para cada uno de los miembros llevando a cabo de una manera mucho más independiente.

  Por otro tanto como parte del estudio efectuando en un capítulo del mismo se destaca el modelo de aprendizaje cooperativo-colaborativo basado en el método de cascada abarcando siete etapas sucesivas y estimulando la capacitación de cada uno de los aprendices. De llevar a cabo el método se deben aplicar un conjunto de normativas a seguir por todo el equipo las cuales son:
  - ✓ Manejar el ambiente de trabajo de una manera segura y confiable manejando cada integrante un alias que lo identifique.
  - ✓ La comunicación entre los miembros se hará a través de correo electrónico.

- ✓ Cada miembro es responsable de una tarea y debe cumplirla como tal ante el grupo de trabajo.
- ✓ Conciencia en el manejo de tiempo sin generar desperdicios de este recurso valioso.
- ✓ Realizar un informe donde se consolide los avances alcanzados por el grupo y darlo a conocer entre sus miembros.

Las etapas a desarrollar son: [18]

1ra Etapa: Conformación del equipo o comisión y asignación de tareas por parte del coordinador.

2da. Etapa: Conformación de grupos de dos personas en donde el uno tomara el papel de responsable de trabajo y el otro de consultor.

3ra. Etapa: Los grupos de trabajo crecen y la responsabilidad que manejan debe ser mayor.

4ta. Etapa: El coordinador asigna un líder a cada grupo el cual debe tener la visión de dirigir el equipo y asignar tiempos en cada actividad que realicen.

5ta. Etapa: Todo el equipo recibe las tareas y el tiempo en que estas se realizaran.

6ta Etapa: El equipo recibe la actividad resuelta en el tiempo propuesto junto con el informe correspondiente.

7ta. Etapa: Es la socialización del trabajo realizado por cada integrante destacando los resultados obtenidos y compartiéndolos a los demás.

- **El teletrabajo en la industria del desarrollo de software en Antioquia**

  Autor: Sergio Andrés Pulgarín Montoya

  Modalidad: Trabajo de Grado - Universidad Pontificia Bolivariana - Programa de Ingeniería Electrónica.

  Año: 2014

  Análisis del estudio:

  En la tesis mencionada como anterioridad se destaca otra posibilidad de trabajar por un objetivo común denominada Teletrabajo en donde sin importar el lugar o el momento donde se encuentren los integrantes de un determinado grupo, puedan alcanzar el mismo propósito y asignar sus actividades sin contar con encuentros presenciales todo el tiempo. Adicionalmente, aunque el estudio

se enfoca exclusivamente en el departamento de Antioquia y el impacto de esta modalidad dentro del mercado, este aporta grandes elementos que definen al teletrabajo como una oportunidad de compartir conocimientos a un determinado proyecto e indagar los posibles parámetros a desarrollar con el fin de lograr el éxito de este. [19]

Por otra parte, y complementando con lo anterior el MINTIC destaca que "De acuerdo con la Ley 1221 de 2008, el Teletrabajo se define como una forma de organización laboral, que consiste en el desempeño de actividades remuneradas o prestación de servicios a terceros utilizando como soporte las tecnologías de la información y la comunicación - TIC para el contacto entre el trabajador y la empresa, sin requerirse la presencia física del trabajador en un sitio específico de trabajo." [20]

Con esto se afirma que el teletrabajo dentro del desarrollo de software es una opción más para las empresas para reducir un gran porcentaje de costos y enfrentarse a una serie de cambios en cuanto a la expansión de su negocio, la internacionalización de los procesos y dar apertura hacia nuevas oportunidades gracias a las ventajas que ofrece la tecnología.

1.2.2 BASADOS EN SEMAT

- **Representación en el Núcleo de SEMAT de prácticas de métodos de desarrollo basados en planes**

  Autor: Ing. Leidy Diana Jiménez Pinzón

  Modalidad: Tesis de Maestría

  Año: 2016

  Análisis del Estudio:

  Este trabajo de grado de maestría resume de manera general el núcleo de Semat y representa algunas prácticas de métodos existentes basados en planes: Rational Unified Process (RUP), Custom Development Method (CDM) y UNC-METHOD bajo este ámbito. De igual manera se detallan las actividades, roles y productos de trabajo propios de cada método, mediante la definición de alfas, espacios de actividad y competencias requeridas. Este estudio sirvió de guía para la presente investigación en la definición de la práctica para equipos distribuidos.

- **Representation of TSP framework into the SEMAT kernel based on the best practices of project management from CMMI-DEV**

  Autores: B. Manrique-Losada; G. P. Gasca-Hurtado & M. C. Gómez-Álvarez

Modalidad: Articulo Facultad de Ingeniería, Universidad de Medellín, Colombia

Aporte del Documento

El estudio anterior propone una relación entre 3 importantes aspectos de la ingeniera de software (TSP, CMMI-DEV, SEMAT) destacando el desarrollo de software en equipos y proponiendo un enfoque alternativo a seguir para lograr con éxito cada una de las fases del proyecto. Adicionalmente la organización de cada uno de los ciclos y de sus actividades realizadas con la mayor disciplina posible contribuye a consolidar el crecimiento del equipo y a establecer buenas pautas que pueden ser implementadas en un momento determinado con la garantía de obtener los mejores resultados. [21]

- **Equipos de desarrollo de software: sus prácticas representadas en SEMAT.**

  Autor: Jorge Orlando Muñoz Rengifo

  Modalidad: Tesis de Maestría de la Universidad Nacional de Colombia Facultad de Ingeniería, Departamento de Ciencias de la Computación y de la Decisión

  Año y lugar: Medellín, Colombia 2015

  Análisis del estudio: [22]

  En este trabajo final de maestría se retoma nuevamente la importancia de crear practicas solidas pero fundamentadas bajo un soporte teórico-científico como lo es SEMAT además de destacar el fracaso que puede existir dentro de un proyecto software como las dificultades que pueden presentarse a lo largo del mismo trayendo consigo grandes pérdidas no solo de dinero sino de tiempo. Adicionalmente resalta que aquellos equipos que tienen un enfoque de autodirección resuelven mejor este tipo de inconvenientes y están más preparados frente a posibles eventualidades, aspecto que define el punto de partida dentro de esta investigación.

  El estudio se hace con el fin de resolver dichos problemas a través de buenas prácticas de software, adaptables y flexibles para cualquier organización que desee utilizarlas y que estas se representen bajo el núcleo de SEMAT.

1.3 SUPUESTOS TEÓRICOS.

1.3.1 Teletrabajo

Actualmente se han puesto a consideración nuevas modalidades de trabajo las cuales han traído grandes pasos a un forma moderna y flexible de laborar; el teletrabajo es una de ellas.

Esta modalidad ha propiciado la posibilidad de que una persona pueda trabajar desde su casa o lugar que desee ajeno a la empresa de la cual haya sido contratada, sin necesidad de movilizarse, generándole a dicho individuo un ahorro en transporte, tiempo y dinero.

Es un modelo ampliamente compartido en beneficios tanto para el empleado como para la organización que lo adopte. Por el lado del trabajador evita su desplazamiento diario, el cual contribuye a disminuir gastos en movilizaciones y a su vez crear un lazo más estrecho con familiares y amigos al tener más tiempo para compartir con ellos; abre la posibilidad de que se usen de mejor manera las tecnologías de la información y comunicación para el desarrollo de actividades y también genera nuevas oportunidades para integrarse al campo laborar para aquellas personas con algún tipo de discapacidad física o psicológica. Por parte de la empresa las ventajas también son bien definidas, comenzando por la que esta puede expandir sus negocios hacia otros rumbos tanto de manera local como regional; puede adquirir personal más capacitado o con mayor experiencia; reduce el porcentaje de costos fijos y la posibilidad de contar con un equipo mucho más robusto donde sus integrantes puede trabajar desde diferentes lugares y horarios. [23] [24]

En Colombia esta modalidad es reciente, sin embargo, ya existe una ley que reglamenta esta condicional laboral y la promueve de manera efectiva. Es entonces como a través del decreto 884 de 2012 que afianza la ley 1221 de 2008 garantizan los mismos beneficios y derechos para aquellos que teletrabajen otorgándoles la posibilidad que puedan desarrollar sus actividades a distancia y apropiar el uso de las Tecnologías de la Información y comunicación como parte fundamental de sus labores cotidianas. [23]

Por otro lado, el porcentaje de teletrabajadores crece con el tiempo, triplicándose ya la cifra que se tenía en el año 2012 que era de alrededor de 30.000 empleados y la cual en 2016 llego a ser ahora es más 96.000 [25]. De igual manera ya son aproximadamente 10.000 empresas las que ya han implementado este modelo, de las cuales un gran porcentaje son dedicadas a la industria de software.

En [26] se puntualiza acerca del impacto del desarrollo de Software a nivel mundial y a su vez se destaca el trabajo remoto, trabajo a distancia, trabajo en línea o Teletrabajo como una posibilidad de gran ayuda para profesionales recién

graduados para que estos adquieran más experiencia, pongan a prueba sus habilidades y destrezas, asuman mayor responsabilidad y disciplina en sus proyectos y tengan mayor flexibilidad en los mismos. Otro aspecto que se resalta es que, aunque es un fenómeno reciente “el mercado de trabajo en línea está creciendo a un ritmo sin precedentes; en donde ha aumentado más del 100% desde 2010”.

1.3.2 Trabajo 3.0

El trabajo 3.0 o también denominado trabajo freelance, muy adoptado por muchas personas hoy en día. Esta modalidad, aunque tiene algunas similitudes con el teletrabajo, se diferencia latentemente en ciertos aspectos. Según una entrevista realizada al director internacional de freelancer.com, Sebastián Síseles [27], en el año 2016 se resalta al teletrabajo como una modalidad de trabajo arraigada a relaciones de dependencia y a un contrato laboral de exclusividad establecido por la empresa, mientras que el Trabajo 3.0 está asociado a laborar de forma independiente para múltiples empleadores de diferentes lugares del mundo sin concurrir a un mismo espacio de trabajo. Eso si ambos modelos incentivan el uso de las TICs como herramientas de apoyo.

Trabajar de manera freelance trae también grandes beneficios. Principalmente los lazos de dependencia hacia uno o varios jefes están totalmente disueltos, ya que existe una mayor libertad en la elección del tiempo de dicho individuo en realizar sus actividades laborales, puedo tener una mayor productividad al no estar bajo tanta presión y la mejora en la calidad de trabajo puede deberse a la armonía de estar en casa, con familiares y/o amigos.

Por el lado del desarrollo de software, trabajar como freelancer es una excelente opción, gracias a las facilidades que ofrece el estar involucrado en más de un proyecto a la vez y obtener un pago remunerado por cada uno de ellos.

Según un estudio de Trabajo 3.0 en Colombia, en el año 2015 se encuesto alrededor de 2500 personas, de las cuales un 22% afirmaron estar involucrados en el desarrollo de software. Este es un porcentaje considerable a diferencia de las otras categorías como: servicios, diseño, multimedia y marketing que se tomaron como objeto de estudio. De dicha población se destaca ante todo la que es joven, pues es la más interesada en efectuar tareas de este tipo. [28]

No obstante, el Trabajo 3.0 puede ser un paso clave en la consolidación de microempresas o de convertir personas en emprendedores otorgando así más posibilidades de empleo.

1.3.3 Crowdsourcing

Esta expresión novedosa hoy en día la cual viene acompañada de dos términos, ‘multitud’ y ‘externalización’. Se trata de un modelo que relaciona el desarrollo de un proyecto manejando patrones de colaboración entre sus participantes. Este tipo de ideas se llevan a cabo de manera voluntaria uniendo diversos puntos de vista para el logro de dicho objetivo común. Esta iniciativa conlleva a la realización de las actividades sin importar que los integrantes no se encuentren físicamente juntos, pero manteniendo fuertes lazos de comunicación. En el desarrollo de software Open Source es una evidencia de este modelo. [29] [30]

1.3.4 Espacios Virtuales para trabajo Colaborativo.

En [31] “los EVTC surgen de la fusión de tres conceptos muy relevantes: Teletrabajo, equipos de desarrollo y espacios virtuales.” Los EVTC pretenden llevar cabo un determinado proyecto el cual congrega un conjunto de tareas las cuales deben ser resueltas de manera específica y de forma grupal, donde cada integrante se encuentra alejado físicamente, pero trabaja colaborativamente con el resto, manteniendo un cierto grado de organización en la realización de cada uno de ellas y lo hace a través de un entorno virtual.

Dicho ambiente será la fuente de interacción entre los participantes, el cual a su vez ofrecerá información en tiempo real sobre las actividades que se están desarrollando y quien o quienes se ven involucrados en su realización.

En cuanto a las ventajas que esta modalidad ofrece, se destaca el obtener una mejor gestión y control del proyecto, manteniendo una secuencia de los avances alcanzados; la reducción de recursos e infraestructura física por el cambio de herramientas tecnológicas, el uso de internet y la computación en la nube; para finalmente obtener una excelente productividad del empleado en el aprovechamiento de su tiempo para circunstancias de índole personal. [31]

1.3.5 Scrum Distribuido

Las metodologías agiles se han convertido en unas excelentes posibilidades para construir software de manera rápida, flexible y adaptable a los clientes al momento de hacer cambios. Scrum es una metodología ágil reciente, muy empleada para el despliegue de software en equipos. Esta se basa principalmente en realizar entregas parciales y regulares del producto final a través de un entorno colaborativo. Debido al despliegue que ha tenido en muchos proyectos, nace la investigación de Scrum Distribuido el cual no es un modelo netamente constituido sino una adaptación de Scrum para contextos en donde los integrantes no comparten una misma ubicación física y/o temporal. [32]

Scrum distribuido busca abarcar ambientes donde la distancia entre los participantes de un proyecto, la diferencia horaria y la falta de sincronismo son retos a superar.

Manejo de Roles con Scrum Distribuido [32]

- **Equipo de Desarrollo.** Con Scrum Distribuido el equipo debe asumir un mayor compromiso con las tareas a realizar y disciplina al momento de ejecutarlas siendo conscientes de que las circunstancias en las que se encuentran dan pie para trabajar con más responsabilidad en busca de alcanzar los objetivos deseados y romper las barreras de comunicación e interacción. A causa del desfase horario presente en algunos integrantes, se sugiere que para la toma de decisiones se efectué en un margen de tiempo amplio, justo para que todos participen de ello y no quede ningún miembro excluido.
- **Scrum Manager.** A parte de su rol normal moderando las reuniones con el equipo, debe encargarse de fomentar una buena comunicación dentro de este, buscando una participación activa y propiciando que el Propietario del producto de las pautas necesarias y la información clara sobre lo que desea.
- **Product Owner.** Dicha persona debe asumir las condiciones propias de este entorno, facilitando una información fluida, hacerlo de manera escrita en un documento que recopile todos los datos de gran ayuda para el equipo sin ambigüedades y definiendo los parámetros a ejecutar.
- **Stakeholders.** En ambientes distribuidos el propietario del producto junto con el Scrum Manager debe tener la suficiente coordinación y comunicación con los stakeholders de tal manera que estos puedan relacionarse en el proyecto con mayor facilidad, facilitando que los sprints se desarrollen de la mejor manera posible, sin ningún inconveniente de acuerdo a las necesidades planteadas desde el comienzo del mismo.

1.3.6 Desarrollo Global de Software

El desarrollo global de software (GSD) es una muestra de lo anteriormente dicho, el cual es muy conocido por orientarse al trabajo distribuido de equipos de desarrollo de software ubicados en diferentes posicionamientos geográficos, caracterizándose por enfrentar desafíos relacionados con distancia, tiempo y cultura. [33] [34]

Adicionalmente el GSD es un fenómeno que ya ha tenido entrada desde hace décadas, muy ampliamente expandido en varios lugares del mundo, pero aun con fuertes problemas por resolverse. De igual manera las ventajas y beneficios que proporciona son realmente notorios, de gran apoyo para la realización de actividades para dichos equipos. [35]

GSD actúa de manera latente en la industria del software el cual viene de la mano con la ejecución de procesos flexibles, que sean aceptadas por el equipo, que con sus experiencias y su conocimiento puedan aportar al crecimiento del proyecto y lógicamente el aprendizaje adquirido sea enriquecedor al relacionarse con personas de otros lugares del mundo.

2. METODOLOGÍA

La metodología que se siguió dentro de la formulación de la práctica de desarrollo colaborativo de software obedece en cierta manera a contribuir a muchas personas que trabajan en diferentes entornos, tiempos y espacios a la organización de sus actividades y adicionalmente aportar a la investigación que se consolida dentro del núcleo de SEMAT y sus propósitos. Las fases efectuadas fueron:

Análisis de los factores y dificultades que se presentan por desarrollar software en diferentes espacios y tiempos.

En esta etapa se buscó consolidar los aspectos que influyen en un equipo de desarrollo cuando este trabaja desde diversos lugares y a diferentes momentos, resaltando las dificultades que pueden surgir, y los factores que influyen en este tipo de circunstancias.

Frente a esto, se hizo en primer lugar una encuesta a equipos de desarrollo presenciales de la Ciudad de Pasto, que compartieron su opinión acerca del tema y lógicamente sobre la forma de trabajo que estos llevan día a día, definiendo así algunos posibles problemas y elementos tanto positivos como negativos que podrían relacionarse.

De igual manera en esta fase del proyecto, se acompañó con las lecturas de documentos bibliográficos y antecedentes relacionados con equipos de desarrollo que trabajan de manera remota.

La consolidación de estos ítems contribuyo a establecer con mayor exactitud los elementos claves a abarcar para la estructuración de la práctica.

Análisis de prácticas y marcos de trabajo existentes aplicables a equipos distribuidos de desarrollo de software

En esta siguiente etapa se extractó elementos muy relevantes del desarrollo de software de manera distribuida, justamente a través del análisis de prácticas relacionadas y aplicables a estos contextos. Dicho estudio facilitó la comprensión mejor del trabajo en equipo cuando se hace a distancia y se identificó los aspectos claves a tratar en la práctica a definir y la cual ira acorde con el núcleo de SEMAT.

Identificación de los roles, actividades y herramientas que harán parte de la práctica para equipos que trabajan en diferentes contextos de espacio y tiempo.

La idea de esta etapa fue que ya partiendo de un estudio del arte sobre los posibles inconvenientes a presentarse dentro de este tipo de desarrollo y partiendo de algunas metodologías existentes en relación con la temática, poder construir la nueva práctica en sus primeros pasos teniendo en cuenta los roles y funciones que van a desempeñarse, las tareas a efectuar y como las tecnologías de la información y comunicación van ayudar a concretar con dicho propósito.

Definición de la práctica para equipos sin restricciones de tiempo y espacio de acuerdo con los roles, actividades y herramientas concretados y a su vez con base en los principios de SEMAT.

Una vez efectuada la identificación de roles, actividades y herramientas, y tomando como punto de partida todos los insumos anteriores se prosigue a consolidar la práctica de desarrollo de software con base en los principios de SEMAT para ser representada mediante su sintaxis gráfica.

2.1 Análisis de los factores y dificultades que se presentan por desarrollar software en diferentes espacios y tiempos.

2.1.1 Acercamiento al contexto mediante lectura de documentación Bibliográfica.

Con el advenimiento del desarrollo de software a distancia y de manera colaborativa, se ha establecido en gran manera como estas circunstancias han tomado un papel trascendental hoy en día. Sin embargo, existen factores que influyen en la organización y realización de las actividades lo que ha propiciado que se originen dificultades que deberán ser manejadas por los participantes.

Cuando se contempla la posibilidad de trabajar remotamente se desglosa una serie de conceptos a relacionarse: interacción, comunicación, tiempo y espacio.

Según la lectura de documentación Bibliográfica en lo que se refiere a la labor de equipos distribuidos se encontró ciertos elementos que contribuyen a determinar cuáles serían las principales dificultades de trabajar de manera remota y los retos que deberían superarse.

Análisis de los retos y dificultades de desarrollar software de manera remota Según Scrum

Scrum como una metodología para el desarrollo ágil de proyectos hace un paréntesis y pretende también buscar una solución para aquellos equipos que trabajan de manera distribuida catalogando a estos como “aquellos en donde los integrantes que pertenecen a este, se encuentran ubicados en diferentes localizaciones en donde no necesariamente comparten un mismo espacio, ni un horario similar, pero si un mismo principio corporativo” [36]. Frente a esto propone a Scrum Distribuido como una adaptación de Scrum para entornos de este tipo. [32]

Las dificultades a superar y la respuesta a dichos retos que Scrum Distribuido expone son:

- **Ausencia de comunicación.** Scrum siempre plantea la comunicación como algo prioritario debido a que sin esta la interacción entre los integrantes sería muy limitada. El pertenecer a un equipo distribuido puede generar dificultades en la manera de comunicarse, debido a diferencias culturales, sociales, de lenguaje o incluso horarias. Frente a esto, Scrum Distribuido plantea que desde comienzos del proyecto se definan los momentos del día en donde el equipo pueda comunicarse, definiendo franjas establecidas que todos puedan respetar y tener en cuenta, empleando las TIC como apoyo a dicho objetivo. [32]
- **Diferencias culturales.** Al trabajar de manera distribuida e interactuando con otras personas y dependiendo de su ubicación geográfica se puede encontrar

que estas posiblemente pertenezcan a una cultura diferente, incluyendo su diversidad étnica, tradicional, y/o religiosa.
Para esto Scrum Distribuido sugiere tomar un espacio de tiempo en el reconocimiento de los integrantes del equipo, detallando así la manera en cómo estos se desenvuelven en este entorno y la manera clara de entenderse unos a otros. [32]

- **Diferencias de lenguaje.** Debido a que los integrantes del equipo pueden pertenecer a diferentes regiones, el lenguaje de cada uno de estos puede variar e impedir la comunicación clara y rápida con el resto de participantes. Como respuesta al reto se debe el uso de un lenguaje común, emplearlo de manera plana, sin jerga o neologismos, ser prudente en el uso de términos para comunicarse, utilizando los más apropiados ya de un lugar a otro puede significar algo muy diferente. [32]
- **Ubicación geográfica y diferencia de horarios.** En función de estos dos elementos según Scrum Distribuido y según este tipo de contextos pueden dividirse los equipos en:
  - ✓ Equipos con horarios flexibles, es decir con horas similares entre los miembros.
  - ✓ Equipos que realizan sus tareas dependiendo de los turnos asignados (mañana, tarde y/o noche).
  - ✓ Equipos que trabajan y comparten un mismo huso horario.
  - ✓ Equipos remotos que no coinciden en un huso horario en común.

  Dependiendo de la franja horaria en que el equipo este sometido va ser más fácil o no aplicar Scrum y su principio de llevar una conversación cara a cara. Para ello podrá tratar de simularse una conversación de manera virtual, con el uso de herramientas tecnológicas como videoconferencias, llamadas telefónicas, chats, etc. [32]

Análisis de los retos y dificultades de desarrollar software de manera remota Según Desarrollo Global de Software

El desarrollo global de software trae en su conjunto una serie de retos que deben tratar de solventarse para que sea un marco de trabajo sólido [37] . Estos desafíos que llevan a sobrepasar una dificultad consecuencia quizá de la falta de organización de tareas, la ausencia de comunicación, la falta de motivación por parte de los integrantes del equipo y demás problemas típicos incluso provenientes del mismo desarrollo de software presencial.

Al manejar este tipo de contextos los retos descubiertos se relacionan principalmente con: [33]

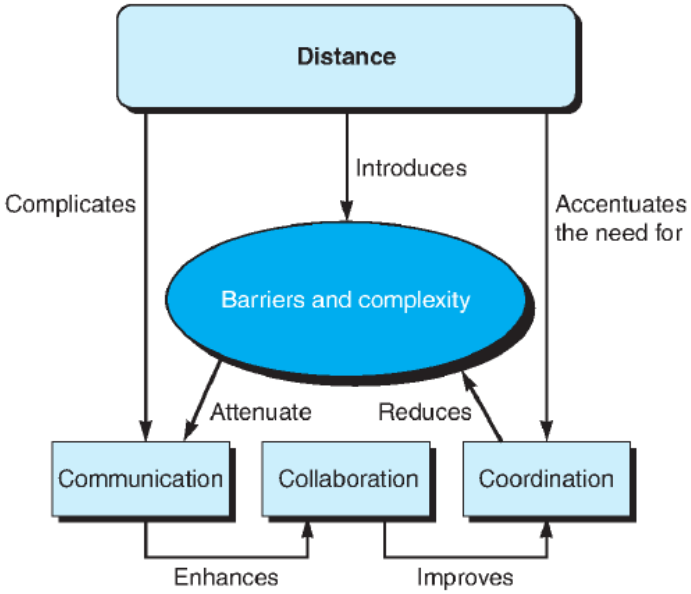
- **Aspectos culturales, étnicos y sociales**: relacionados con la manera de pensar y transmitir una idea, raíces ancestrales, tradiciones, la cual difiere del lugar proveniente de cada individuo.
- **Comunicación eficaz**: Es muy común que dentro de los equipos de desarrollo la comunicación se vea influenciada por algún tipo de error a raíz de la mala transmisión de la información ocasionado que esta llegue de manera incompleta o incoherente. Este elemento es de vital importancia manejarse puesto que contribuye a la toma de decisiones de manera adecuada y a mantener lazos efectivos de armonía entre los participantes. Además, la comunicación debe llevarse en un lenguaje estándar sin ningún tipo de neologismos o vocablos propios de cada región.
- **Distribución y asignación de actividades de manera digerible para sus participantes**, evitando así alguna inconformidad en el desarrollo de las mismas y delegando roles de acuerdo a las habilidades y destrezas de los mismos.
- **Direccionamiento del proyecto hacia caminos de éxito**, manteniendo el control al cambio y la visibilidad de funcionamiento en todo momento.
- **Interacción con otras organizaciones en el desarrollo de proyectos**, incentivando así a la participación activa entre las mismas.
- **Derechos de propiedad intelectual**, principalmente respaldando el apoyo y el aporte que la comunidad desarrolladora realice al proyecto, teniendo en cuenta los fines legales que acojan en cada país.

Según el PhD. Ricardo Colomo Palacios profesor en el Departamento de Informática de la Universidad Carlos III de Madrid en una conferencia acerca de desarrollo global de software en el año 2014 resalto algo muy importante, “Todos los problemas que hay en el desarrollo de software normal se heredan en el desarrollo global de software de manera clarísima.” [34], lo que conllevaría a inferir que el trabajar de manera presencial trae consigo ciertas dificultades que fácilmente son heredadas al trabajo remoto. Consecuentemente se resalta que desarrollar software desde diferentes lugares e influenciados por la atemporalidad es un fenómeno que día con día está tomando fuerza y es común en las organizaciones pero que también prepara a estas a derribar ciertas barreras: distancia geográfica (posicionamientos), distancia temporal (diferencia de usos horarios), distancia cultural (interacción humana). [34]

Además, en [38] se profundiza acerca de los factores humanos como una parte fundamental en el desarrollo de software, resaltando el papel de los equipos globales. En esta sección se afirma como estos equipos se han ido constituyendo alrededor del mundo y que además de poseer las características típicas de un equipo de desarrollo de software tradicional también necesitan bastante cuidado en factores tales como la comunicación, la coordinación, la colaboración y la toma de decisiones, siendo el ultimo el más difícil de manejar debido a la complejidad del problema software a manejar, el riesgo que se corre al tomar una decisión y

los diferentes puntos de vista que se someten para llegar a una acuerdo de solución.
A continuación, se muestra una relación de dichos aspectos en el marco de los equipos globales. (véase Figura 1)

Figura 5. Factores que afectan a un equipo global de software



Fuente: [39]

La anterior fuente determina un enlace entre dichos aspectos, resaltando la necesidad de trabajar bajo estos criterios para reducir en cierta manera las barreras que el GSD conlleva.

Continuando con el análisis de la documentación bibliográfica, el artículo [40] nos menciona alrededor de su contenido, un estudio realizado en la identificación de riesgos en entornos GSD, mencionando 39 de ellos los cuales consolidan un marco de trabajo para propiciar un cambio en el área organizacional.

El estudio de dichos factores se dividió en seis categorías las que según el autor menciona son: "*Task – Actor"* enfatizado en la posición y las capacidades que tiene el individuo frente a la realización de sus actividades; *"Task - Structure"* hace referencia a la función de la organización en encaminar correctamente la realización de labores; "*Task-Technology"* en el fuerte lazo de las herramientas tecnológicas en el desarrollo de actividades*; "Actor-Structure"* en cuanto a las expectativas que tiene el individuo de su organización para motivarlo y trabajar de la mejor manera posible; *"Actor-Technology"* en como el individuo usa dichas herramientas para sacarle buen provecho y utilidad; *"Structure-Technology"* en

cómo la empresa acoge en su estructura organizacional el uso de la tecnología. [40]

Factor Tarea - Actor

En esta categoría se relacionan siete factores de riesgo del más al menos relevante según la investigación realizada por los autores de [40], comenzado por las *diferencias lingüísticas* que se refiere a los neologismos propios que tiene cada región y lo cual puede provocar malos entendidos entre las personas del equipo; *la diferencia en conocimientos* que pueden poseer los individuos frente a un tema u otro provocando incomodidad e insatisfacción; la *comunicación fluida* entre los participantes y la *ausencia de mantener una conexión en tiempo real* con ellos para tomar decisiones y que su vez se presente el riesgo de *no realizar correctamente la tarea asignada* produciendo cambios muy poco significativos en el proyecto. Por último, se resalta a la *deficiente consolidación de documentación* de lo efectuado y *la poca apropiación y experiencia del individuo* en trabajar bajo este contexto.

Factor Tarea – Estructura

En esta sección se detalla más sobre los riesgos que pueden existir entre los procesos de la organización con el desarrollo de actividades, de los cuales se extrajeron para este estudio 7 de los 17 en total encontrados de acuerdo su nivel de relevancia.

- Riesgo 1. El trabajar separados unos de otros debido a las distancias geográficas que se tienen influye en que la coordinación y el control del proyecto se vuelva un tanto complejo. [40] [41]
- Riesgo 4. La calidad en los procesos se vuelve un tanto critico en este tipo de actividades, lo que atribuye a que este factor no se tome en cuenta como se debería. [40] [41]
- Riesgo 5. Al no tener tantos encuentros cara a cara las relaciones interpersonales se debilitan ocasionado dos alternativas, que se atribuya un gasto adicional en un viaje para encontrarse con el equipo esporádicamente o simplemente acudir a video llamadas lo que generaría que la comunicación se vuelva un poco más fragmentada. [40] [41]
- Riesgo 6. La formulación de objetivos dentro del equipo debe ser integral y clara en todos los sentidos para evitar que haya algún tipo de malinterpretación y pérdida de tiempo por efectuar alguna tarea de la forma incorrecta. [40] [41]
- Riesgo 8. Los roles que se asignen deben ser de acuerdo a las capacidades y destrezas que tenga cada individuo. [40] [41]
- Riesgo 9. Ausencia de compromiso por trabajar en equipo bajo estas circunstancias. [40] [41]

- Riesgo 17. El mal manejo de la propiedad intelectual, frente a las licencias o asuntos legales que pueda conferir puede generar algún tipo de conflicto. [40] [41]

Factor Tarea – Tecnología

Esta categoría trae dos factores descritos pero muy estrechamente relacionados entre sí. El uso de la tecnología y de las herramientas puede variar de acuerdo al nivel de conocimiento y experticia que cada individuo tenga en el manejo de las mismas. Además, es muy probable que la cobertura en su uso en algunas partes sea limitada o incluso restringida. [40]

Factor Actor – Estructura

Esta categoría relaciona ante todo como el comportamiento y la cultura del individuo puede influenciar en el desarrollo de tareas. Los principales riesgos que se mencionan son:

- Riesgo 1. Al poseer el individuo su propia cultura, su manera de expresión, su terminología propia, esta puede ocasionar malentendidos sino se tiene un previo conocimiento de ella. [40]
- Riesgo 3. Al no tener un correcto apropiamiento de la cultura del otro, las relaciones interpersonales se pueden afectar y adicionalmente generar inseguridad en la toma de decisiones. [40]
- Riesgo 4. Al carecer una interacción cara a cara entre los integrantes del equipo la confianza entre los mismos puede verse sesgada. [40]
- Riesgo 6. Al tener una comunicación más limitada, el entusiasmo por trabajar en equipo y el colaborarse mutuamente puede quedar por desapercibido. [40]

Factor Actor – Tecnología

Esta sección enfrenta los riesgos que puede haber cuando un usuario maneja una herramienta sin la debida experiencia, y como el entrenamiento que se debe tener junto con la colaboración en aprender unos de otros en el uso la misma, debe ser una prioridad para evitar cualquier inconveniente. [40]

Factor Tecnología – Estructura.

Finalmente, en la última categoría se abarca riesgos relacionados con la selección de una mala estructura con una tecnología existente o viceversa. El más destacable aspecto es de la deficiente comunicación para elegir las herramientas, provocando incluso que no se elijan las más pertinentes y esto pueda provocar efectos secundarios afectando principalmente la información que se esté administrando. [40]

Lo anteriormente dicho permitió analizar más ampliamente los factores que influyen en el ámbito GSD dentro de una organización lo que contribuye a esquematizar la situación y buscar enfrentar dichos retos de la mejor manera posible.
El articulo [42] recalca barreras como factores de éxito durante la mejora en el proceso de desarrollo de software en contextos GSD. En este sentido complementa la posibilidad de construir software manejando criterios importantes desde una buena planificación, una correcta implementación de la calidad y al menor costo posible.

De las barreras que los autores del texto citaron se destaca para la investigación:

- La ausencia de recursos en donde se expone que, si no se hace un presupuesto idóneo de los elementos a necesitar y una asignación correcta de los mismos, en el proyecto va ser muy difícil obtener los resultados deseados.
- Otro ítem a discutir es que un personal sin experiencia o sin conocimiento en pro de trabajar colaborativamente puede provocar la mala realización del trabajo y provocar caer en fracasos consecutivos.
- La necesidad de tener una metodología formal que promueva la mejora de los procesos de desarrollo de software en entornos GSD e incentive la consolidación de estrategias que contribuyan a la ejecución apropiada de tareas.

Los aspectos claves por los cuales pueden contribuir positivamente a estas situaciones son:

- Promover la coordinación, comunicación y colaboración entre los integrantes del equipo para que la implementación de software process improvement sea practica y eficaz en GSD.
- Asignación de recursos viendo los límites del proyecto y manejando siempre una buena planificación.
- Establecimiento de metas de acuerdo a su nivel de relevancia y que a su vez sean realizables.
- Incentivar al equipo a trabajar colaborativamente y promover sus habilidades mediante entrenamiento, capacitaciones.
- Intercambio de información, lo que contribuye al aprendizaje continuo de todos los participantes, a promover la colaboración y a tener más elementos que aporten al conocimiento colectivo.
- El éxito de un proyecto se mide también con los lazos fuertes que entre los integrantes de un equipo hayan construido, permitiendo así mayor confianza al momento de trabajar, una opinión más abierta frente a un determinado problema y una mayor conciencia para ser resuelto.

Es así como el desarrollo de software distribuido además de que se enfrenta a tres factores reales como es la distancia, el horario y la cultura, también presenta dificultades (mostraron anteriormente) que deben ser tratadas para que un gran

porcentaje puedan ser solventadas de la mejor manera posible. A su vez [15] expone que también hay aspectos positivos que esta forma de desarrollo trae como el de que empresas adquieran y conozcan del talento de otras personas del mundo, la reducción de costos fijos sea relevante y lógicamente una gran posibilidad de acercamiento a clientes que probablemente también se encuentren lejos del equipo de desarrollo.

Análisis de los factores que afectan al desarrollo de software dependiendo de la interacción con el equipo de trabajo

Vale la pena resaltar que dependiendo de la magnitud y complejidad del proyecto software que se tenga es como el equipo debe buscar la mejor forma de trabajar e integrase colaborativamente por alcanzar dicho ideal.

El equipo debe fomentar una participación activa en todo aspecto, cultivando siempre la posibilidad de enlazarlo con un ambiente integral y bastante cooperativo. Roger Pressman [15] define a un equipo con características como la confianza, el talento y la unión promoviendo así un ajuste de habilidades y destrezas aptas para trabajar colectivamente.

Sin embargo, a pesar de la persistencia por lograr los objetivos, existen muchas adversidades que ponen al equipo a superarlas de la mejor manera, circunstancias que pueden fracturar la relación y el entorno, pero que deben ser tratadas a tiempo para que no produzcan efectos colaterales.

- Mala asignación de roles y responsabilidades, lo que generaría que no se distribuya bien las funciones de cada persona.
- Ambiente desentendido, en donde cada quien hace lo que quiere sin tener en cuenta la valoración del otro para una posible sugerencia o mejora.
- No se tienen estrategias que controlen la posibilidad de un fallo de un integrante del equipo por cualquier tipo de eventualidad.
- Falta de motivación en el equipo para promover el talento y la participación en la decisión de elegir libremente la tarea en la cual se sienta más cómodo y pueda llevarla a cabo con un alto grado de satisfacción.
- Carencia de técnicas que alimenten la comunicación y coordinación entre los participantes.
- Mala administración de la información concerniente al proyecto, ocasionando el desconocimiento en ciertas fases o la ignorancia en la funcionalidad de cierto componente.

Cabe aclarar que, en condiciones remotas entre los integrantes del equipo, estas situaciones pueden influir aún más en cada actividad que se realice y pueden provocar que el desempeño y rendimiento de estos sea cada vez menor si no se toma las medidas necesarias para aliviar dichos problemas.
Además en [15] se evidencia un conjunto de autores que exponen los problemas más frecuentes en desarrollo de software distribuido, destacando principalmente a

Sutherland quien agrupa dichas dificultades en cinco categorías tales como son las *"técnicas"* enfatizadas en el manejo estándar de los datos y herramientas a involucrar; *"estratégicos"* en cuanto al manejo, asignación de recursos y actividades en estos entornos, *"culturales"* referentes a la interacción con los comportamientos, etnias y tradiciones propias de cada individuo; *"comunicacionales"* enfocadas a la expresión abierta de ideas unos a otros y finalmente de *"seguridad"* aplicables a la confidencialidad en el manejo y transferencia de información.

Es importante recalcar que cuando los equipos se encuentran en situaciones geográficamente dispersas, la comunicación es el principal ítem a manejar tal y como le demuestran varias fuentes [37] [32] [15] [42] , destacando que entre más alejados estén los integrantes más difícil será mantener el contacto entre ellos: "la gente ubicada a 100 metros o más, tiene apenas el 5% de chance de hablar con otra" [43] , aspecto que requiere bastante cuidado y se debe remediar de la mejor manera haciendo uso de las herramientas TIC más pertinentes a cada circunstancia.

2.1.2 Acercamiento al contexto mediante realización de una encuesta a equipos de desarrollo de software

Una de las prioridades de esta fase, fue inicialmente hacer un acercamiento a los desarrolladores de software que trabajan de manera presencial.

El objetivo principal de este estudio era el de determinar la importancia que tiene para los miembros del equipo de desarrollo de software, el trabajar conjuntamente sin ninguna restricción de tiempo y espacio.

Para esto se efectuó una encuesta de once preguntas donde la población objeto de estudio fueron centros, empresas y equipos de desarrollo de Software de San Juan de Pasto, de las cuales se seleccionó como muestra, de manera intencional aquellas que quisieron colaborar con dicho propósito las cuales se listan en la tabla Número 1.

Tabla 1. Resultados Listado de empresas y centros de desarrollo de software participantes.

Equipos de desarrollo	Recuento	Porcentaje
Centro de Informatica Udenar	16	25,8%
CJTYT	13	21,0%
Institución Universitaria Cesmag	8	12,9%
GRIAS	6	9,7%
Apptividad	4	6,5%
Hashito Apss	4	6,5%
Parquesoft	4	6,5%
JHWeb Pasto	2	3,2%
Otras	5	8,1%

Fuente: Este estudio.

El instrumento se aplicó durante el periodo septiembre – octubre del año 2017, contando con la participación de 62 desarrolladores en total.
Al ser un muestreo no probabilístico de tipo intencional se tuvo en cuenta criterios de aceptación en cuanto a la selección de los centros y/o empresas de desarrollo como:

- Que cuenten con personas que hayan desempeñado labores en desarrollo de software.
- Que tengan la disposición por colaborar en el objetivo del estudio.
- Que manifiesten algún tipo de interés por la investigación.
- Que quieran apoyar voluntariamente a proyectos de índole académica.

Para el análisis de los resultados obtenidos tras efectuar la encuesta, este fue de manera cualitativa y univariado descriptivo.

La encuesta se dividió en tres secciones, la primera enfocada en el estado actual del encuestado dentro de la empresa. La siguiente enfocada en la interacción con el equipo de desarrollo de software y finalmente la última dedicada a las diferentes modalidades de trabajo.

La solución de dicho instrumento por parte de los desarrolladores de software permitió en primer lugar establecer el nivel profesional que poseen, determinando así su experiencia en la construcción de software. Además, se pudo establecer cómo es la interacción del desarrollador con el resto del equipo y los posibles problemas que se han presentado. Finalmente se identificó si dicha persona ha trabajado de manera remota en labores de esta área, así como los beneficios obtenidos.

Resultados del estudio

Frente al estudio efectuado se abarcó el análisis de tres aspectos importantes tales como dificultades presentadas durante el proceso de desarrollo de software, ventajas y desventajas de trabajar de manera remota.

a) Dificultades en el desarrollo de software

Claramente es evidente que los problemas dentro del proceso de desarrollo de software siempre van a ser notorios, los cuales si existen de manera presencial claramente se va presenciar en circunstancias remotas.

En la encuesta realizada a los equipos se postularon 5 posibles opciones, las cuales fueron obtenidas tras la lectura de documentación bibliográfica.

- **Falta de comunicación con el equipo de desarrollo**. Relacionada con la participación activa del equipo manifestando sus puntos de vista y respetando el aporte del otro.
- **Falta de organización en la asignación y/o ejecución de las actividades.** Se enfatiza en como las tareas son llevadas a cabo dentro del equipo, si hay un orden en las asignaciones de las mismas a los participantes y un seguimiento en la realización de estas.
- **Mala estimación de tiempo para el desarrollo de software.** El cual es un problema muy común y se presenta debido a que el equipo de desarrollo no sabe estimar desde inicios del proyecto la cantidad de recursos que va a emplear incluyendo el tiempo que va invertir en cada una de las actividades que este involucre y los posibles contratiempos que pueden producirse.
- **Cambio constante de requerimientos dentro de proyectos software.** Es muy complicado desde un principio definir requisitos de un proyecto software sino existe una comunicación abierta y clara con el cliente frente a las necesidades y expectativas que posee de este.
- **Uso inapropiado de metodologías o modelos de desarrollo de software.** El cambio de metodología de trabajo entre los equipos puede causar retrasos en los proyectos e incluso el fracaso total de este.

De la información recolectada se obtuvo los resultados siguientes.

Tabla 2. Dificultades en los equipos de desarrollo de software según encuesta.

Dificultades	Recuento	%
Mala Estimación	44	71,0%
Cambio de Requerimientos	41	66,1%
Falta de Organización	23	37,1%
Falta de Comunicación	16	25,8%
Uso inapropiado de las metodologías	15	24,2%
Otra	5	8,1%

Fuente: Este estudio.

Figura 6. Dificultades en los equipos de desarrollo de software

Fuente: Este estudio.

Los resultaron arrojaron que la mala estimación de tiempo para el desarrollo de software es el mayor problema presentado con un porcentaje del 71%, seguido por el cambio constante de requerimientos durante el proyecto con un 66,1%. Por otra parte, un 8.1% de la población opino que otras de las dificultades podrían ser la ausencia de un buen presupuesto que contemple de manera adecuada los recursos económicos a emplear dentro de un proyecto software; el poco conocimiento de trabajar en equipo y la toma de decisiones de forma apresurada.

b) Ventajas de desarrollar software de manera remota

El desarrollo de software de manera remota y la deslocalización en este tipo de eventos trae consigo una seria de aspectos positivos y negativos a tratar. Del total de los encuestados, 27 de ellos correspondiente al 43,5% de la muestra respondieron que alguna vez han optado por trabajar de manera remota.

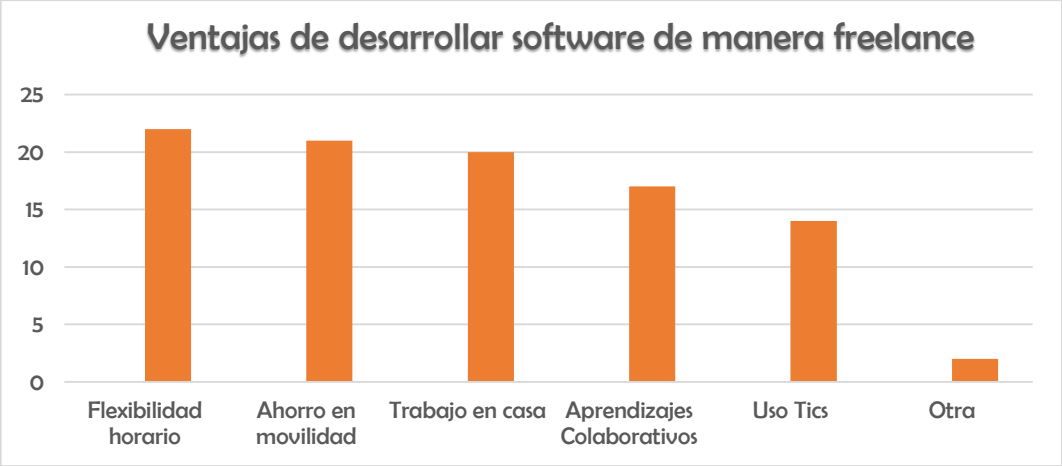
En base a lo anterior entre las ventajas que ellos consideraron fueron:

Tabla 3. Ventajas de desarrollar software de manera remota según encuesta.

Ventajas	Recuento	%
Flexibilidad horaria	22	81,5%
Ahorro en movilidad	21	77,8%
Trabajo en casa	20	74,1%
Aprendizajes Colaborativos	17	63,0%
Uso Tics	14	51,9%
Otra	2	7,4%

Fuente: Este Estudio.

Figura 7. Ventajas de desarrollar software de manera remota según encuesta.



Fuente: Este estudio.

La pregunta anterior para los encuestados fue de opción múltiple, lo que permitió tener su punto de vista con más de una posibilidad en consideración a este aspecto.

La flexibilidad de horarios una de las ventajas que encabeza la lista propiciando una mejor distribución del tiempo para el desarrollador usándolo tanto para espacios de trabajo como para asuntos personales. La siguiente ventaja es el ahorro en movilidad lo que genera adicionalmente un ahorro en dinero, demostrando que el trabajar de esta manera propicia nuevas oportunidades para optar por esta modalidad dentro del desarrollo de software.

c) Desventajas de desarrollar software de manera remota

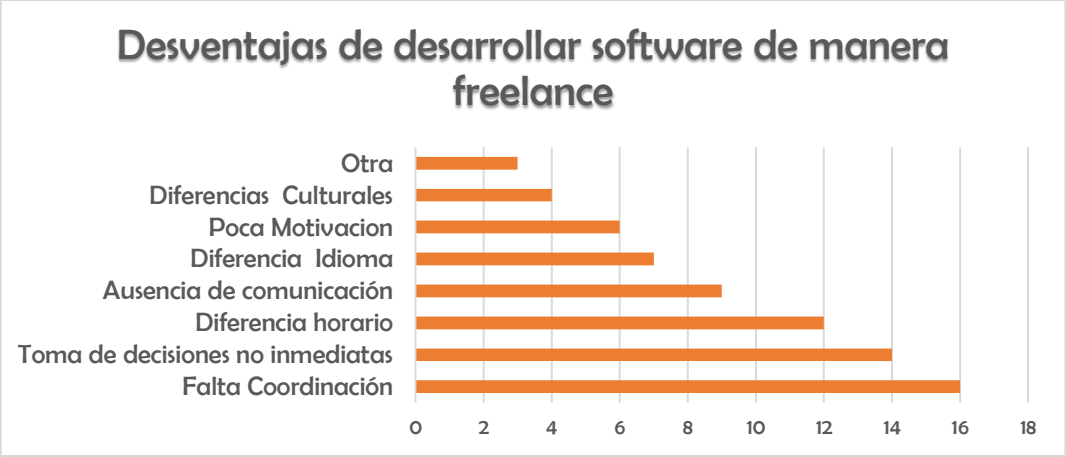
De igual manera se indagó por las desventajas que consideran al aplicar esta modalidad.

Tabla 4. Desventajas de desarrollar software de manera remota según encuesta.

Desventajas	Recuento	%
Falta Coordinación	16	59,3%
Toma de decisiones no inmediatas	14	51,9%
Diferencia horaria	12	44,4%
Ausencia de comunicación	9	33,3%
Diferencia Idioma	7	25,9%
Poca Motivación	6	22,2%
Diferencias Culturales	4	14,8%
Otra	3	11,1%

Fuente: Este estudio.

Figura 8. Desventajas de desarrollar software de manera remota según encuesta.



Fuente: Este estudio.

En este caso la falta de coordinación dentro del equipo de desarrollo de software es uno de los problemas que encabeza la lista, demostrado que sin una buena organización de cada una de las etapas del proyecto se podría poner en riesgo el funcionamiento del mismo e incluso provocar la dificultad de comprender y asimilar las ideas a construir por parte de los participantes.

Otro ítem a considerar es la toma de decisiones no inmediatas, ya que al trabajar de manera remota es un tanto difícil que se decida por hacer algo prontamente, esperando la participación de todos los integrantes más aún cuando estos se encuentran en diversas zonas horarias y lo que generaría a su vez retrasos en el proyecto.

2.2 Análisis de prácticas o marcos de trabajo existentes aplicables a equipos distribuidos de desarrollo de software

Según [15] se denomina desarrollo distribuido o disperso de software cuando los integrantes del equipo tienen muy pocas o casi ninguna posibilidad de hacer encuentros presenciales durante el proyecto debido a las circunstancias mencionadas con anterioridad.

En esta segunda fase de la investigación se pretende indagar por prácticas o metodologías existentes que abarquen el desarrollo distribuido de software para personas que trabajen colaborativamente desde diferentes posicionamientos geográficos y horarios.

2.2.1 Prácticas de desarrollo de software tradicionales que contribuyen al desarrollo de software de manera remota

Para iniciar esta etapa es de vital relevancia resaltar algunos elementos que son aplicables en cualquier contexto de desarrollo de software. En [15] se menciona que para lograr el éxito de un proyecto se debe comenzar por efectuar una buena planificación que agrupe todas sus extensiones tanto de recursos, campo de acción y ambiente de trabajo. La formulación de métricas adecuadas, indicadores de gestión de calidad y una buena estimación de riesgos a tiempo, ayuda a controlar de mejor manera cada una de las tareas por realizarse evitando caer en fracasos consecutivos. A continuación, algunas de las practicas por detallar:

- **Práctica en la administración y testeo de su código por parte del desarrollador.** En esta práctica se resalta la particularidad en el desarrollador al mismo tiempo que vaya haciendo su código vaya preparando sus pruebas para determinar si hay fallas en el mismo y evitar inconsistencias futuras, lo que atribuiría a manejar en su trabajo con un alto grado de calidad.
- **Empleo de un modelo descentralizado – controlado dentro del equipo.** Es una práctica que orienta a realizar las actividades formando equipos más pequeños de tal manera que cada uno tenga su propio un líder el cual mantendrá una comunicación vertical con el jefe de todo el proyecto quien tendrá al mando la administración y regulación de las operaciones principales del mismo.
- **Manejo de un paradigma sincronizado.** En donde las tareas se simplifican a lo menos posible para reducir en gran porcentaje la comunicación entre los participantes.

2.2.2 Prácticas adoptadas por organizaciones para trabajar de manera distribuida

Con el avance de las tecnologías y la aparición de internet, las puertas para que muchas empresas puedan trabajar de manera distribuida con diversas personas del mundo demuestra el impacto que este tipo de modalidad ha traído hoy en día.

- **Siemens Corporate Research.** Es una organización que ha trabajado sus proyectos de manera deslocalizada [15]. Su labor se traduce a una serie de pautas para evitar la menor cantidad de riesgos y fracasos posibles. En virtud de dicho planteamiento se resalta que es mucho más difícil manejar proyectos en donde todas sus fases se hacen de manera distribuida ya que el control y la coordinación pueden verse afectadas de manera directa. Frente a esto, ellos proponen un trabajo arduo, bien estructurado por el líder del equipo, en donde deben estar completamente definidas las etapas de ingeniería de requisitos y análisis y donde las funciones asignadas a los participantes estén en base a sus capacidades.

- **SirsiDynix.** En [15] se habla de SirsiDynix una empresa que ha venido trabajando desde la década de los noventa con la implementación de software para la gestión servicios bibliotecarios**.** Por otra parte, el trabajo que realizan lo hacen de manera deslocalizada, aplicando como prácticas las estrategias planteadas por Scrum Distribuido resaltando las reuniones diarias virtuales entre jefes y demás integrantes para discutir los avances del proyecto, un acercamiento entre el cliente y el líder mucho más abierto, reutilización de código e implementación adicional de prácticas propuestas por XP para su mejor desempeño.

- **Yahoo.** Considerada una organización muy reconocida hoy en día, dedicada a ofrecer un portal global de medios a sus usuarios [44]. Ha llevado a cabo proyectos de manera distribuida tales como *Yahoo Podcasts* el cual contó inicialmente con integrantes de California (diseñadores y arquitectos de software principalmente) pero ya viendo la necesidad de recursos y presupuesto más tarde se decide interactuar con equipos de Bangalore, India de manera conjunta [45]. Para esta intención, seleccionaron a Scrum como la metodología a usar dentro del equipo, resaltando que en un principio para la realización de los encuentros cara a cara virtuales la situación era complicada al tener un desfase de doce horas entre las ciudades. Los más afectados eran los residentes de la India puesto que las reuniones para ellos solían ser por las noches, momento por el cual veían interrumpido sus encuentros familiares para dedicarse a su trabajo. Este inconveniente produjo serios roces entre los equipos por lo cual implementaron una regla llamada *"Go Local"* [45] en donde las reuniones se hacían ya de manera local, sin tener que requerir de la presencia del otro equipo y la comunicación solo se limitaba más a llevarla a cabo con los líderes de cada zona respectivamente. Adicionalmente el horario

de conversaciones en línea fue alternado para que ambas partes puedan uniformidad de condiciones. Finalmente, al instaurar dichas estrategias la implementación del Manifestó Ágil fue satisfactoria y los enlaces de comunicación entre los participantes se habían fortalecido cada vez, lo que género que se convirtiera en un caso de estudio exitoso.

Otro proyecto de Yahoo que fue trabajado de igual manera fue Vespa News [45]. Este por su lado manejó equipos de California y de Noruega con una diferencia horaria de nueve horas. Los problemas que enfrentaron fueron ante todo la jornada laboral puesto que para los noruegos esta finaliza diariamente a las 5 pm mientras que para los californianos su horario es más extensivo y flexible. Sin embargo, sus reuniones se orientaron primero en capacitarse colectivamente para garantizar significativamente avances del proyecto.

Por último, esta Yahoo answers [45] como una iniciativa que logro unir el esfuerzo de muchas personas de diferentes continentes y valorar su aporte colaborativo. Todo comenzó debido a que la plataforma fue realmente aceptada por el público y a consecuencia de esa demanda necesitaba ser escalable. A razón de ello el proyecto se consolidó con la ayuda de desarrolladores de Sunnyvale, Oficinas de California y Londres, Reino Unido.

- **Xebia Software**. Caracterizada por ser una organización que implementó el desarrollo distribuido en los países bajos y en la India aplicando en su metodología prácticas de Scrum y XP [46]. Esta forma de trabajo se lleva a cabo de manera organizada, proponiendo charlas diarias entre participantes de tal manera que se integren más armónicamente a las tareas cotidianas y se sientan cómodos con el resto del equipo. Además, la productividad del proyecto es en gran parte satisfactoria gracias a una medición de los costos en historias de usuarios y una asignación pertinente de actividades junto con una retroalimentación continua de los avances logrados. Un caso real de evidencia de dicho panorama fue en la ejecución del Proyecto Xebia ProRail PUB el cual era un software que se encargaba de la administración de información relacionada con horarios para viajeros del sistema férreo de Holanda.

  El desarrollo de dicho aplicativo trajo consigo la distribución del sistema debido a la complejidad del mismo y al manejo en tiempo real que se requiera de todo ese conjunto de datos. La labor que se realizo fue ardua pues necesitaba pronto que este prosperara y ofreciera al cliente escalabilidad y alto desempeño. Prontamente el trabajo era bien designado en ambos equipos de las dos regiones y se alinearon con respecto a prácticas, estándares, herramientas y roles. Como desafíos que presentaron fue las diferencias culturales, el compartir el contexto y la forma de trabajar usando ante todo herramientas de comunicación como Skype y otras para el almacenamiento del código en repositorios y finalmente la realización de trabajo de manera local.

- **SAP, TCS y LeCroy.** Este conjunto de empresas trabajó colectivamente por la consolidación de proyectos de manera distribuida manejando 22 practicas

divididas en cinco categorías: *coordinación entre los sitios* relacionado con la interacción de los individuos y la motivación que se debe propiciar para que estos cumplan su trabajo con satisfacción ; *herramientas y tecnologías apropiadas* que permitan un alto grado de comunicación, colaboración y por supuesto que posibiliten la idea de estar conectados todo el tiempo; *vínculos sociales* en donde los enlaces armónicos se construyan fuertemente y que a través de capacitaciones se contribuya a llevar a cabo de la mejor manera los quehaceres dentro del ambiente de trabajo; intercambio de conocimiento que propicien compartir nuevas ideas y aprender de estas experiencias; y *gestión de componentes* y su reutilización evitando invertir más tiempo o costos innecesarios. [15]

- **Organización global que mantiene centros de desarrollo de Software en América y Asia.** La experiencia de esta organización trae la reunión de talento de dos continentes en donde siguieron procesos de estándares internacionales como lo son el Modelo de Madurez de Capacidades (CMM), PM Cuerpo de conocimiento (PMBoK), el marco de trabajo de Microsoft Solutions (MSF) y la metodología Scrum [47]. Manejaron dos tipos de proyectos, siendo el primero enfocado en el dominio financiero y el cual conto con la participación de ocho personas ubicadas tanto en Norteamérica como en Suramérica. El segundo proyecto más orientado a la parte logística el cual integra a personas de América y de Asia involucrando a países como Estados Unidos, Brasil y China. Entre los problemas que enfrentaron fue principalmente el carecer una estructura adecuada dentro del equipo, la ausencia de una documentación que los respalde tanto en la parte de requisitos como en la realización de las demás fases, la mala planificación de tareas y las dificultades en comunicarse lo cual afecta el compromiso para llevar a cabo una buena integración y una mutua colaboración.

  Entre las sugerencias que ellos plantean fue que tanto los casos de uso como las historias de usuarios deben estar integradas para evitar inconvenientes en la interpretación y sincronización de lo que se vaya hacer. Además, el líder del proyecto debe ser una persona que mantenga el orden dentro del grupo y capaz de lidiar con las tareas que a todos competen, priorizarlas y mantener un cuadro de tiempo estimado para cada una de estas. Otro aspecto es que cada negociación que se llegue en cada reunión que se haga, se mantenga documentado y se tenga soporte de ello. Lógicamente promover que tanto el desarrollo como el testeo estén de la mano para identificar posibles fallas a tiempo y velar por la calidad del código fuente es una recomendación de vital relevancia. Finalmente, la implementación de una herramienta que contribuya a la planificación y ejecución de las actividades contribuirá muchísimo al control de estas y propiciaría a la toma de decisiones más asertivamente posible. [47]

- **BMC Software (Identity Man- unidad de negocio en gestión).** Esta organización también contó con la aplicación de Scrum dentro de dos tipos de casos: integrantes de equipos distribuidos ubicados en diferentes países, y equipos completos totalmente deslocalizados teniendo participación de Tel Aviv, Francia y Florida y también donde los propietarios del producto también se encontraban dispersos. En [48] se resalta que el uso de Scrum ayudo significativamente a llevar a cabo lo planeado de la mejor manera posible, aunque aún existieron dificultades que se deben remediar en cuanto al flujo de información que se debía llevar a cabo del equipo a sus líderes.

- **Thought Works.** Esta empresa reúne un conjunto de prácticas agiles del esfuerzo realizado en cuatro años de experiencia en países como la India, América del Norte y Europa para equipos distribuidos. Algunas de las practicas que ellos propusieron fue el de realizar visitas de carácter presencial de vez en cuando para mantener la integración entre las partes, usar repositorios como wikis para el almacenamiento de la información, efectuar reuniones netamente cortas para tener una idea de lo que se está llevando a cabo, y lógicamente mantener una documentación pertinente y actualizada que sea accesible para todos. [15]

- **Proyecto Blender.** Blender es una herramienta para la creación 3D de modelos, animaciones, simulación y demás composiciones multimediales aptos para juegos y videos, siendo de código abierto y de uso gratuito. [49]
  La versión de Blender 2.8 es un proyecto de software el cual se lleva a cabo con muchos desarrolladores de varios continentes y su colaboración se hace en línea a través de correos electrónicos, comunicación vía chat, etc. El equipo de Blender al estar distribuido por todo el mundo tiene muy pocas oportunidades de reunirse cara a cara, sin embargo, es en eventos como Blender Conference o Blender Development Workshops donde la manera de interactuar presencialmente unos a otros se llevan a cabo [49]. Se espera que haya más contribuyentes a lo largo de esta labor y que la versión beta esté lista para mediados del año 2018.

2.3 Identificación de los roles, actividades y herramientas que harán parte de la práctica para equipos que trabajan en diferentes contextos de espacio y tiempo.

La práctica de desarrollo de software que se empieza a formular ya en su primera instancia, pretende abarcar exclusivamente la etapa de codificación o desarrollo tal y como se expone en la delimitación de la investigación. La etapa de ingeniería de requisitos, el análisis y diseño no se abarcará dentro del estudio puesto que se recomienda al igual que en [37] estas se hagan lo más cercanas al cliente y no de manera distribuida para estar sujeta al control de cambios que pueda presentarse.

2.3.1 Elementos para la Integración de la práctica.

La práctica al ser un marco de trabajo para desarrolladores de software pretende ser aplicada en el contexto donde los participantes colaboren en el proyecto, pero no necesariamente de manera presencial. Es por eso que se ve la importancia de tener en cuenta los actores y actividades que cada uno realice, así como también las herramientas que pueden ser aptas para este tipo de entornos.

Frente a los problemas mencionados en etapas anteriores se puede resaltar que esta práctica pretende buscar una solución para dificultades consideradas para la presente investigación como las más relevantes y las cuales fueron agrupadas en tres categorías:

- **Problemas con la planificación del trabajo en equipos distribuidos.** Relacionada con la asignación de responsabilidades, el uso de herramientas con la debida instrucción, y la manera de llevar a cabo las actividades de manera organizada siendo un trabajo a distancia y quizá con desfase horario lo cual impediría la toma de decisiones de manera inmediata.
- **Problemas de comunicación.** Aplicable en la interacción entre participantes y las diferencias lingüísticas entre ellos.
- **Problemas de acoplamiento entre componentes de software.** Enfatizado a la coordinación, control y acoplamiento que debe existir al momento de construir un componente software. Se puede decir que es el motor principal para que en la etapa de codificación sea exitosa cuando esta se distribuye totalmente ya que de lo contrario esto podría ocasionar problemas mayores como pérdida de tiempo, repetición del trabajo, confusión y hasta malos resultados.

2.3.2 Elementos claves para trabajar.

En la investigación se ha definido que el producto software a desarrollar dentro de un proyecto se dividirá de acuerdo a:

Funcionalidades. Conjunto de características que hacen que el software sea apto para satisfacer una necesidad expuesta por un cliente y que esta funcione en base a unos requisitos establecidos. Se relacionan a las generalidades que tiene el sistema.

Procesos. Conjunto de actividades que se realizan de manera interrelacionada para lograr un objetivo y que contribuyen a consolidar las funcionalidades del sistema.

Componentes. Es un elemento parte clave de un proceso del sistema el cual tiene una función específica a realizar y donde un colaborador participara en su desarrollo.

2.3.3 Roles

Tabla 5. Roles de la práctica.

Roles	Descripción	Objetivo
Cliente	Persona(s) que expone una necesidad de proyecto que busca ser solventada mediante la aplicación de conocimiento en áreas de Tecnologías de la información y desarrollo de software.	Exponer una serie de requerimientos al líder del proyecto y al equipo para que esta sea trabajada y solucionada a través de un producto software.
Líder del proyecto	Persona a cargo de la totalidad del proyecto que maneja un alto grado de liderazgo, empatía con el resto del equipo y fuertes estrategias para promover un excelente desempeño en las actividades a realizar.	Dirigir el proyecto de software frente a los jefes de zona y colaboradores con alto grado de responsabilidad, partiendo de unos requisitos otorgados por un cliente.
Jefe de Zona	Persona capaz de propiciar el control, seguimiento de actividades y que opere como nodo de control dentro de la zona asignada al fin de mantener el orden y la coordinación necesaria entre los participantes.	Mantener un fuerte enlace de comunicación tanto con el líder del proyecto para garantizar la regulación y la adecuada implementación de cada una de las funcionalidades del sistema como con los colaboradores para la pertinente asignación de actividades.

Colaboradores	Grupo de personas que interviene directamente en el proyecto y que con su conocimiento aportan líneas de código dentro del mismo de manera significativa.	Propiciar una relación participativa con el jefe de zona y trabajar colaborativamente con el resto del equipo.

Fuente: Elaboración propia del autor.

2.3.4 Responsabilidades de acuerdo con los roles

Líder del Proyecto.

- Proveer a los jefes de zona la documentación necesaria relacionada con el estudio de ingeniería de requisitos, análisis y diseño pertinente al proyecto de manera entendible y sin ambigüedades.
- Transmitir información clara y precisa sobre las funcionalidades a realizar dentro del sistema a través de un documento formal que así lo establezca.
- Exponer la arquitectura y el diseño del sistema a los jefes de zona.
- Proporcionar el estándar de trabajo a los jefes de zona el cual deberá incluir los lenguajes de programación y plataformas a usar, la estructura y presentación del código fuente, así como las entradas y salidas que se deben obtener por cada proceso que se haga.
- Disponer de un tutorial o manual de indicaciones sobre las herramientas a emplear dentro del desarrollo del proyecto para que exista homogeneidad en su uso y manipulación.
- Establecer las metas del proyecto, la estimación pertinente a su cumplimiento y un seguimiento para su realización exitosa a través de los jefes de zona.
- Mantener una comunicación permanente con los jefes de zona para tener un reporte diario de lo que sucede en cada región con el logro de actividades y poder retroalimentar dichos procesos con todos los participantes.
- Encargarse de la gestión del proyecto proporcionando soluciones efectivas en todas las áreas y resolviendo inquietudes siempre que exista una dificultad.
- Determinar un cronograma estimado para la realización de cada uno de los procesos establecidos.

Jefes de Zona

- Mantener una comunicación constante con el líder del proyecto.
- Fomentar comunicación entre los integrantes del equipo en esa zona y fortalecer la toma de decisiones de manera activa y participativa.
- Incentivar a que sea el desarrollador el que escoja la actividad a realizar a medida de sus competencias y habilidades, motivándolo a hacer sus quehaceres con satisfacción y empeño.

- Complementar la información de los componentes a desarrollar proporcionada por el líder del proyecto con las pautas necesarias para que el colaborador lo implemente de la mejor manera posible.
- Suministrar la información correspondiente al componente asignado a cada colaborador.
- Asignar horarios de encuentros virtuales entre los integrantes de la zona.
- Facilitar toda la información proporcionada por el líder del proyecto en cuanto a los estándares de codificación, de manejo de herramientas y de documentación que se deben seguir.
- Propiciar la reutilización de código fuente entre los colaboradores.
- Manejar entregas parciales y finales que se hagan de cada componente software para tener un control y organización de los avances realizados.

Colaboradores

- Llevar a cabo la actividad que haya sido seleccionada en base al estándar de trabajo expuesto por el jefe de zona.
- Trabajar colaborativamente con el resto del equipo en miras de hacer lo mejor para el proyecto.
- Mantener una relación abierta y comunicativa con el resto de los colaboradores y con el jefe de zona.
- Aportar con código fuente a la realización de alguna actividad dentro del proyecto.
- Trabajar el componente del software con el o los lenguajes de programación establecidos y en las herramientas proporcionadas por el jefe de zona.
- Implementar componentes, localizar y corregir defectos

Cliente.

- Exponer la necesidad que requiera ser solventada al líder del proyecto.
- Establecer una comunicación permanente con el líder del proyecto para ver los respectivos avances hasta su cumplimiento total.

2.3.5 Herramientas

- **Herramientas para comunicación**

  a) Slack. Es una herramienta de trabajo colaborativo que permite la interacción entre integrantes de un equipo, mejorando así la forma de comunicación que estos manejen. Slack permite crear canales directos entre los participantes los cuales contribuyen para discutir temáticas o tareas específicas en grupos [50] . Es multiplataforma ofreciendo su servicio tanto para dispositivos móviles como para escritorio permitiendo así que puedan usar la herramienta con mayor accesibilidad.

Ventajas

- ✓ Tiene integrado Dropbox lo que permite compartir documentos e información en tiempo real.
- ✓ Slack suple con las necesidades de los usuarios al contar con enlaces directos de aplicaciones como Hangouts (mensajería instantánea), Asana (gestión de tareas en trabajo colaborativo), Jira (administración de tareas y organización flujos de trabajo), Papertrail (gestión de registros), Pingdom (comercio electrónico), Pivotal tracker (administrador de proyectos), Buildbox (creación de juegos), Mailchimps (marketing por correo), Trello (administrador de proyectos), Github (repositorio) entre otras.
- ✓ Ofrece la posibilidad de enviar mensajes bien sean de manera privada o pública.
- ✓ Permite enviar extractos de código fuente en diferentes formatos.
- ✓ Su versión gratuita es muy buena permitiendo buscar y navegar por los 10.000 mensajes más recientes, cuenta con 5GB de almacenamiento, 10 servicios integrados y sin límite de miembros [51]. También cuenta con su versión de paga la cual obviamente ofrece más beneficios.
- ✓ Comunicación más divertida y espontanea con el uso de emojis.
- ✓ Permite personalizar notificaciones, perfil del usuario, colores dentro de Slack al gusto de cada integrante y modificar los permisos respectivos.
- ✓ Slack no sólo permite crear grupos, conversaciones privadas, o diversos canales, compartir enlaces o archivos adjuntos, sino que es un servicio altamente configurable. [52]

Desventajas

- ✓ Solo la versión de pago permite hacer llamadas por medio de los canales.

Figura 9. Interfaz de Slack

Fuente www.slack.com

b) Stride. Es una herramienta de mensajería que permite la comunicación con el equipo gracias a los servicios que proporciona.

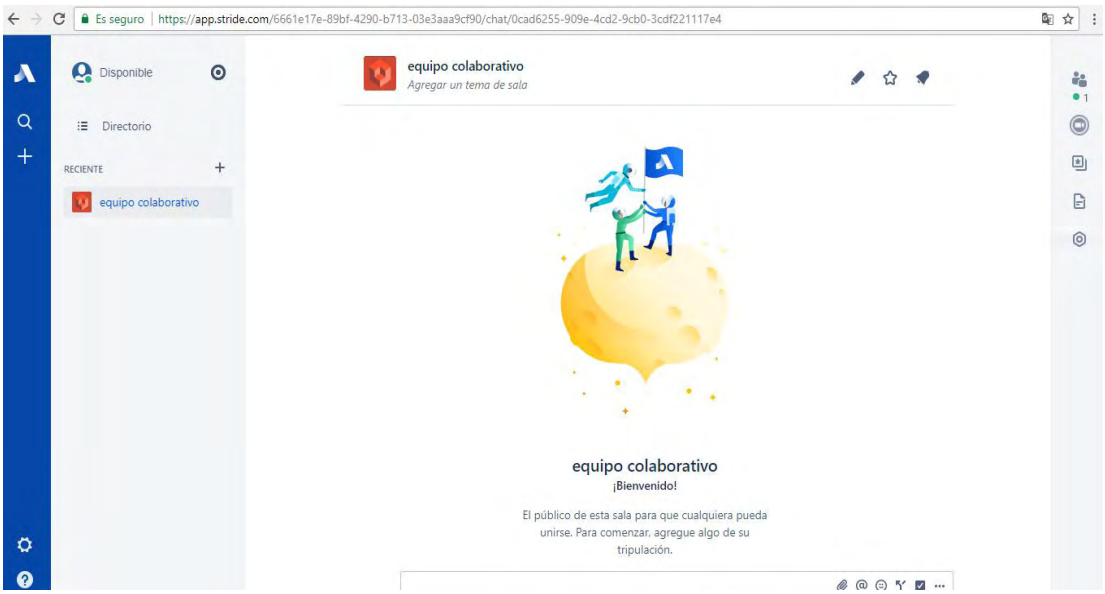
Ventajas

- ✓ Cuenta con Chat grupal y mensajería directa; voz y videoconferencia y herramientas de colaboración integradas.
- ✓ Agiliza y facilita la comunicación en entornos de trabajo.
- ✓ Tiene con un tablero de actividades donde se puede resumir el estado del proyecto, los sprints realizados, informes, entregas y cualquier tipo de componente dentro del mismo.
- ✓ Permite compartir escritorios en tiempo real.
- ✓ Permite compartir archivos, se integra con servicios de terceros (como pueden ser los servicios de almacenamiento en la nube Dropbox y Drive), así como utilizar aplicaciones y bots en los chats. [53]

Desventajas

- ✓ Es de versión de paga, aunque posee una versión limitada gratuita el gratuito con tan solo 5GB de espacio para compartir archivos.

Figura 10. Interfaz de Stride



Fuente. https://app.stride.com/

c) Skype. Es una herramienta que permite la comunicación entre personas a través de llamadas telefónicas, video llamadas, o mensajes de texto de manera instantánea. Muy empleada hoy en día ya que es adaptable a cualquier tipo de necesidad y lista para ser usada en teléfonos móviles, PC o tableta.

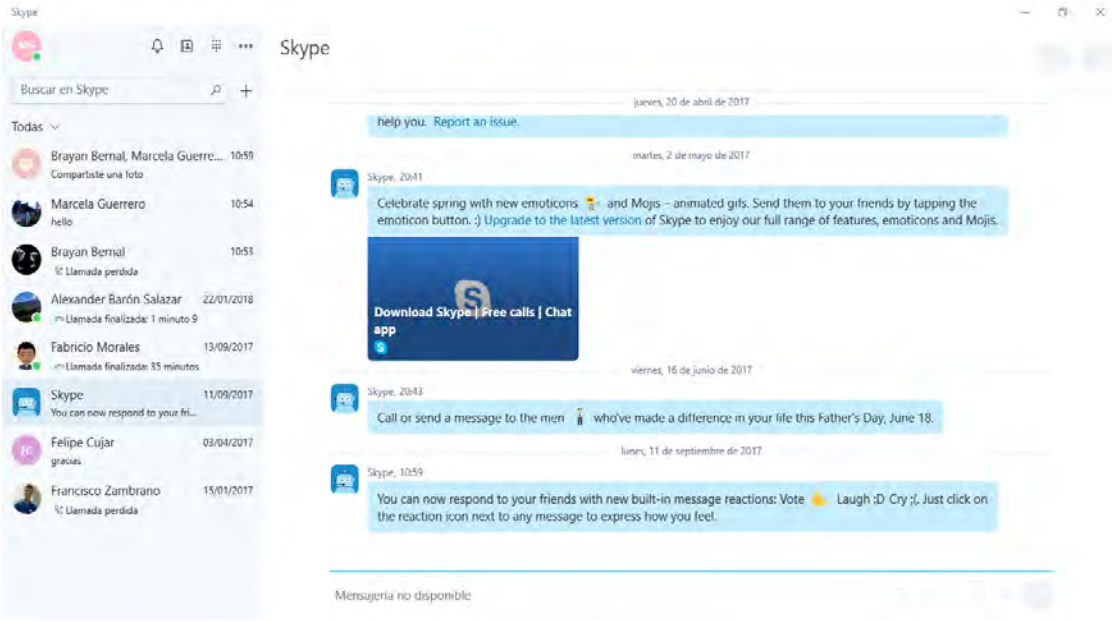
Ventajas

- ✓ Se descarga fácilmente y su uso es gratuito.
- ✓ Pueden los usuarios enviar y compartir archivos de una manera fácil y rápida.
- ✓ Envío de mensajes (texto o de video) de manera inmediata.
- ✓ Permite compartir escritorio en tiempo real.
- ✓ Realización de conversaciones grupales y hasta videollamadas de este tipo.

Desventajas

- ✓ Se necesita pagar para acceder a servicios adicionales como llamar a teléfonos y enviar mensajes SMS.

Figura 11. Ventana de Skype



Fuente. Skype

- **Herramientas para la gestión de actividades**

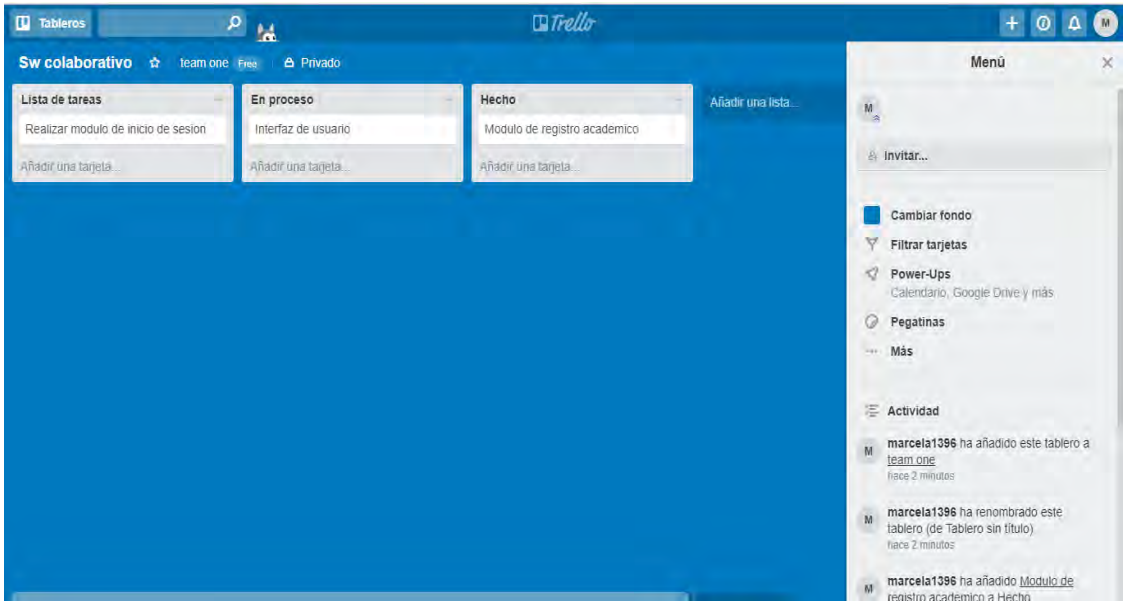
a) Trello. Es un software que permite la gestión de proyectos muy utilizado para el trabajo colaborativo entre equipos de manera rápida, gratuita y flexible. Permite administrar el flujo de proyectos con mayor facilidad, organizándolo y midiendo su progreso.

Ventajas

- ✓ Permite añadir checklists, etiquetas, fechas de vencimiento y otros elementos.

- ✓ Sincronización con dispositivos móviles eficientemente.
- ✓ Proporciona un tablero de tareas el cual permite medir el desempeño de estas y su estado actual.
- ✓ Trello es una herramienta de gran importancia que facilita la comunicación fluida entre los integrantes de un equipo ofreciendo además sincronización en tiempo real.
- ✓ El tablero de Trello permite mantener la información del proyecto de manera organizada a través de sus listas ubicadas de manera horizontal el cual contiene cards las cuales pueden arrastrarse, soltarse o reordenarse. Dichos elementos muestran la sección de tareas, imágenes, archivos adjuntos, fechas de entrega, etiquetas de colores y comentarios de otras personas que interactúen en el tablero teniendo así acceso a dichos datos. [54]
- ✓ Es gratuito.

Figura 12. Tablero de Trello



Fuente. https://trello.com/

b) Jira Software. Es una herramienta altamente empleada por organizaciones la cual está diseñada para proveer un servicio fácil y ágil en todas las etapas de creación de productos software ya que permite llevar a cabo una adecuada planificación y supervisión de las actividades para el logro de excelentes resultados.

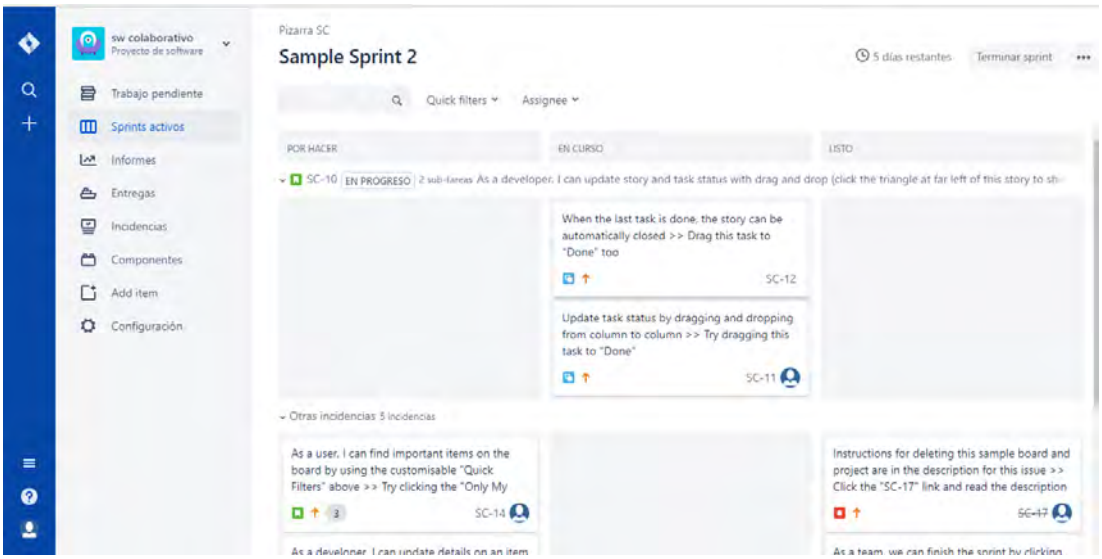
Ventajas

- ✓ Permite crear informes para que los participantes estén al tanto del progreso de las tareas y cuenten con el panorama del proyecto en tiempo real.
- ✓ Pueden visualizar el flujo de trabajo que se está llevando a cabo.
- ✓ Cuenta con la integración de diversas herramientas destacándose Confluence (software de colaboración de contenidos para trabajo en equipo), Bitbucket (Sistema de control de versiones distribuidas), Bamboo (servidor para la integración continua, el despliegue y la entrega de software) y Stride (comunicación para equipos).
- ✓ Jira está enfocado para el trabajo en proyectos de desarrollo ágil como Scrum, Kanban o metodologías mixtas permitiendo así que se pueda tener una mejor estimación y eficacia al momento de construir software.
- ✓ Jira Software ofrece un nivel superior de transparencia sobre el trabajo de cada equipo y mantiene el sincronismo en la organización. [55]

Desventajas

- ✓ Jira es un software privativo y de versión de paga para poder acceder a todos sus servicios.

Figura 13. Jira Software



Fuente. https://es.atlassian.com/software/jira

c) Basecamp. Es una herramienta apta para la gestión de proyectos permitiendo que la información esté disponible en cualquier momento y de manera organizada.

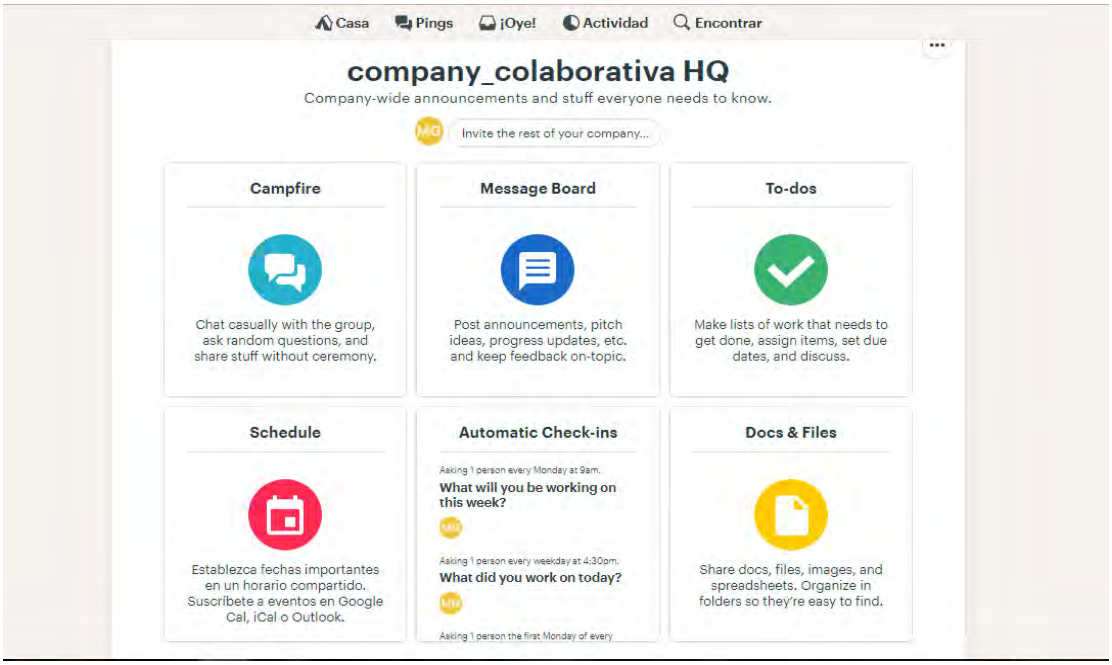
Ventajas

- ✓ Centraliza la información manteniéndola de forma clara.
- ✓ La pantalla de inicio de Basecamp le ofrece una vista donde puede observar todo lo que sucede dentro de su organización.
- ✓ La ventana de Basecamp proporciona seis importantes herramientas: Tareas pendientes para el trabajo de seguimiento, un Tablero de mensajes para publicar anuncios y actualizaciones, una sala de chat para conversaciones rápidas e informales con el equipo, un Calendario para publicar fechas límite, Documentos y Archivos para organizar todos los activos y notas que todos deben hacer su trabajo, y Check-ins automáticos para obtener información del equipo de forma regular. [56]
- ✓ Basecamp proporciona una mejor organización del trabajo estableciendo fechas de vencimiento y asignando responsabilidades.

Desventajas

- ✓ Basecamp es de versión de paga, puede probar su funcionamiento por 30 dias.

Figura 14. HQ de Basecamp



Fuente. https://basecamp.com/how-it-works

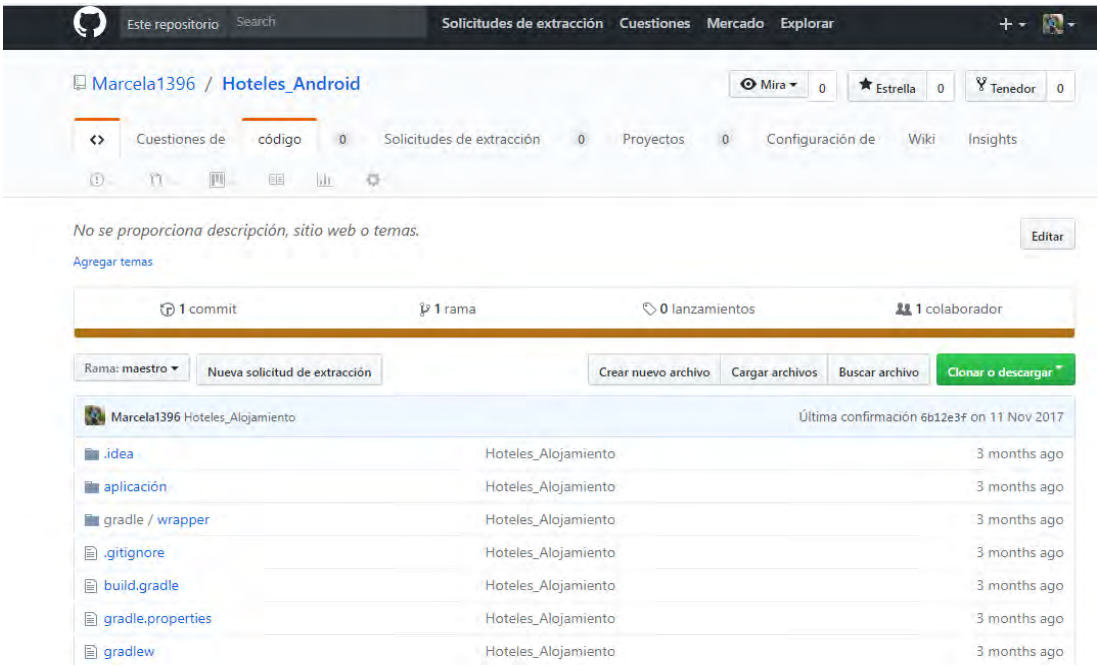
- **Herramientas para la administración de repositorios**

  a) Github. Es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. Este servicio permite alojar el repositorio de código además de que brinda herramientas útiles para el trabajo en equipo, dentro de un proyecto. [57]

  Ventajas.
  - ✓ Ofrece herramientas adicionales para el trabajo en equipo como lo son las wikis para el manejo de versiones; un sistema de seguimiento de problemas que permiten tener al tanto a los miembros de las dificultades que pueden surgir en el software y proponer sugerencias a ello; elementos para revisión de código con sus respectivas anotaciones y un visor de ramas para medir los progresos realizados a lo largo del proyecto.
  - ✓ Github hace el trabajo en equipo más ágil y sencillo, ayuda a la detección de fallos, a disminuir errores humanos, al seguimiento por etapas del proyecto y al mantenimiento de diferentes entornos. [58]
  - ✓ El servicio pone a disposición de todos los usuarios repositorios de código públicos y libres sin límites.

  Desventajas
  - ✓ El manejo de repositorios privados está sujeto a una versión de paga.

Figura 15. Github



Fuente. https://github.com/

b) Bitbucket. Es un sistema de control de versiones distribuidas que facilita la colaboración con su equipo a través de Git.

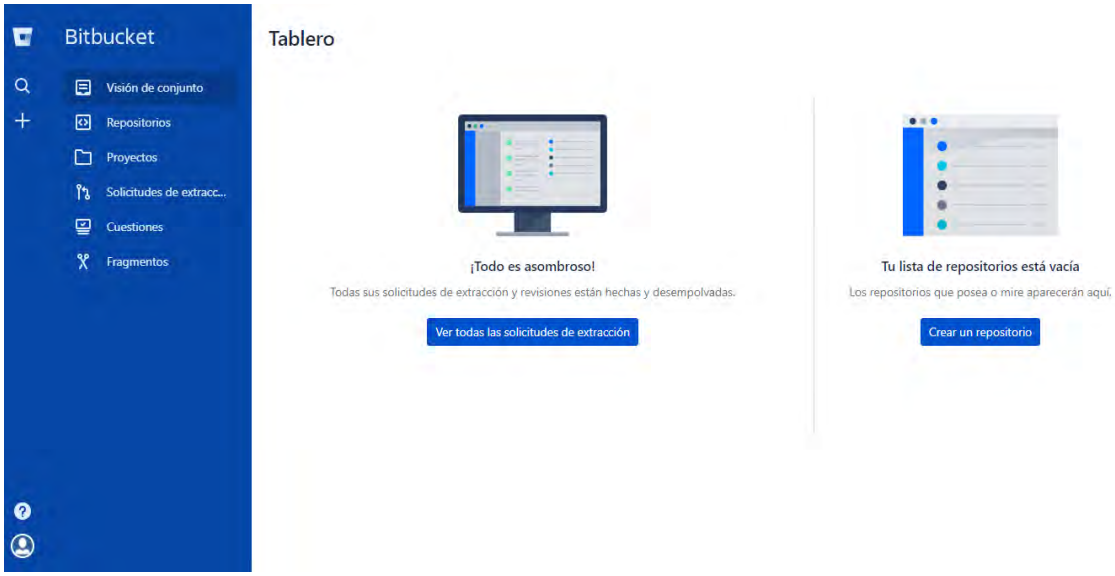
Ventajas

- ✓ Ofrece alto rendimiento y escalabilidad en el manejo de versión de código fuente.
- ✓ Apto para trabajar en cualquier tamaño de equipo gracias a los modelos de implementación flexibles que ofrece.
- ✓ Permite manejar repositorios de tipo privado y público de manera ilimitada.
- ✓ Garantiza almacenamiento de archivos grandes de Git especialmente para aquellos que son dedicados a la creación de juegos.
- ✓ Asegura el control del flujo de trabajo mediante el control de permisos a los repositorios.
- ✓ Facilita la integración con otras herramientas de vital relevancia como Trello, Jira, Bamboo y Hipchat.
- ✓ Usa Git pero también es compatible con Mercurial.

Desventajas.

- ✓ Bitbucket permite en su plan gratuito crear repositorios privados y públicos indistintamente, sin límite de proyectos a alojar, pero limitado en el número de colaboradores permitidos para el caso solo serán hasta 5. [59]

Figura 16. Bitbucket



Fuente. https://bitbucket.org/

2.4 Acercamiento inicial a la Propuesta

En búsqueda de la solución de los problemas planteados se identifica que, al distribuir el trabajo en un equipo al momento de desarrollar un software, se necesita de un alto grado de coordinación en todo el grupo para manejar eficientemente cada una de las tareas por realizar.

2.4.1 Definición de las actividades.

Las actividades que se muestran a continuación relacionan la construcción de software en equipos que trabajan de manera distribuida abarcando principalmente la organización de estas y el acoplamiento de los componentes durante esta fase. Sin embargo, es importante establecer que en este tipo de circunstancias es necesario consolidar un equipo que se colabore entre sí y puedan lograr las metas propuestas sin problema alguno. El objetivo principal de esta investigación es netamente abarcar la etapa de desarrollo, sin embargo, se sugiere de manera general una forma de efectuar la conformación de equipos para la realización de tareas.

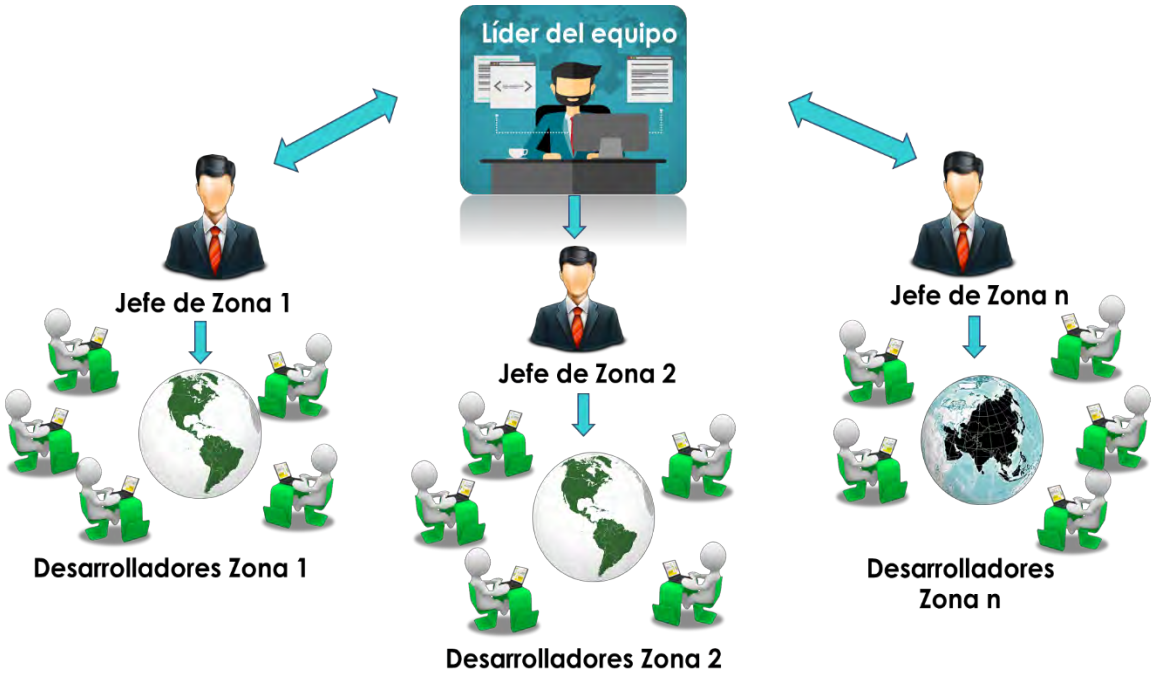
Generalidades en la Conformación de equipos distribuidos.

El líder del proyecto tendrá a cargo un grupo de personas que serán los encargados de realizar la codificación de manera distribuida. Dentro de ese grupo se establecieron dos tipos de roles: los jefes de zona y los colaboradores.

El jefe de zona será aquel que estará cargo de un grupo de desarrolladores en un rango de horario similar y se encargará conjuntamente con ellos de sacar adelante un determinado proceso del sistema. Sera una persona que además de tener habilidades en programación, tendrá la facilidad de liderar a dicho subgrupo y encaminarlo a trabajar de manera colaborativa, propiciando la comunicación e interacción entre los mismos. Además, mantendrá una relación mucho más cercana al líder para medir la evolución del proyecto. Para el caso de los colaboradores serán personas que estarán a cargo de un jefe de zona y trabajaran aportando líneas de código al componente por el cual les fue otorgado.

Como los participantes estarán trabajando desde diferentes localidades y quizá con diferencia horaria, la idea es consolidar subequipos de mínimo dos personas. Tal si fuera por desfase horario, la conformación se hará con participantes donde la diferencia no supere las tres horas. Esto contribuirá a que puedan tener contacto frecuente entre ellos, disponiendo de algún espacio para interactuar y trabajar más a gusto. Si no hubiese dicho desfase, hacerlo de preferencia por ubicación geográfica, lo cual no descartaría el encuentro presencial si las posibilidades así lo permitiesen. En la figura 17 puede visualizarse la arquitectura de la práctica.

Figura 17. Arquitectura de la práctica



Fuente: Elaboración propia del autor.

Actividades para el Desarrollo de Software en Equipos Distribuidos.

- **Definición de las funcionalidades, procesos y componentes del sistema.** El proyecto de software se dividirá en base a funcionalidades, procesos y componentes. Las funcionalidades describen esas necesidades que el sistema debe satisfacer para que el cliente este conforme con el trabajo que se realice. Dichas funcionalidades permitirán identificar los procesos del sistema los cuales contienen los componentes que cada colaborador implementará.
- **Detallar las variables de entrada y salida que cada componente tendrá para propiciar el acoplamiento de estos con el respectivo proceso.** A través de la plantilla de definición de componentes es donde el líder del proyecto establecerá claramente la dimensión del componente que se va a desarrollar y la prioridad en que se requiera. Además, es allí donde se debe especificar tanto los parámetros de entrada que recibirá como los parámetros de salida que cada uno de estos deberá retornar, las conexiones con bases de datos, si las tuviera, las interrelaciones con algunas dependencias, entre otras características que permitan que la integración con el resto de componentes sea fácil, rápida y oportuna.
- **Especificar el modelo y diseño de cada componente.** Tal y como se había descrito cada componente aparte de poseer la información respectiva que detalle claramente su objetivo y especificaciones, debe contar con un modelo,

diseño, bosquejo, y/o diagrama que se debe anexar, con el fin de que el colaborador comprenda más fácilmente lo que debe hacer y no cometa ciertos errores al momento de codificar.

- **Establecer los lenguajes de programación y plataformas de desarrollo a emplear y especificar la forma de presentación del código fuente.** Un estándar de trabajo puede definir el grado de organización que se tiene al momento de desarrollar software. Es evidente que, al trabajar de forma distribuida, el acoplamiento de entre componentes y con sus respectivos procesos puede ser un factor de sumo cuidado ya que es posible se genere conflictos si las salidas arrojadas no son las esperadas, es por eso que el líder del proyecto debe proporcionar toda la información acerca de las plataformas a manejar, lenguajes de programación, forma de presentación y organización del código para lograr la homogeneidad requerida. De igual manera los jefes de zona deberán dar a conocer al resto de colaboradores de este tipo de información para que el trabajo sea digerido por todos de la misma manera.
- **Proporcionar el estándar para el manejo de documentación por cada componente.** La documentación es muy importante en cada componente software que se esté realizando puesto que servirá como guía de manejo para usuarios finales. Es por eso que el líder del proyecto debe dar a conocer dicho propósito con el fin de estandarizar la formar de documentar cada aporte que se realice y mantener la uniformidad en este aspecto.
- **Asignación de procesos a las zonas consolidadas.** Una vez se cuenta con toda la información referente a lo que el equipo en su totalidad va a desarrollar, el líder del proyecto transfiere a cada jefe de zona un proceso el cual incluirá los componentes que se codificaran. Este será quien distribuya entre sus colaboradores dichas tareas que a su vez contribuirán a lograr de manera conjunta el objetivo deseado.
- **Asignación de componentes a los colaboradores de la zona.** Cada colaborador se encargará de un componente a la vez, el cual será efectuado por selección propia, sintiéndose a gusto de implementarlo y para pueda trabajar competentemente en este. Si hubiese dos o más personas interesadas en hacer el mismo componente, será el jefe de zona el que vera quien es la persona más idónea para realizarlo de acuerdo a las habilidades en desarrollo de software que posean.
- **Revisión del modelo y la plantilla de información del componente.** En esta actividad, el jefe de zona entrega a cada colaborador la información respectiva al componente que va a desarrollar, junto con los diseños, modelos o diagramas que se relacionen con el fin de que este conozca claramente todo lo que involucra.
- **Codificación del componente.** El colaborador una vez comprendido todo lo que debe realizar el componente, este procede a codificarlo en base a las pautas anteriormente establecidas en los estándares (forma de presentar las líneas de código, lenguaje de programación indicado) y lógicamente en

relación a las entradas y salidas que debe emitir de acuerdo a la plantilla de definición del componente que se le facilito.

- **Reutilización de código fuente.** La colaboración dentro del equipo es indispensable para facilitar un ambiente agradable entre los interesados y debe estar presente durante todo el periodo de desarrollo de software. La reutilización de código fuente debe ser una prioridad para trabajar dentro del proyecto, así como el conocimiento que puedan compartir para hacer posibles mejoras o sugerencias.
- **Entregas parciales del componente de software.** El jefe de zona estimará un tiempo aproximado en la realización de cada componente y cada desarrollador estará de acuerdo en si este es pertinente para hacerlo. Cada componente tendrá unas tareas en específico las cuales deberán visualizarse a través de una plataforma (herramienta) donde los participantes de dicha zona estén al tanto de dicho seguimiento y podrán colaborarse unos a otros si ven la necesidad de hacerlo. Los resultados en la realización de cada componente se irán midiendo de acuerdo a como estos sean entregados al jefe de zona, en base a los insumos iniciales solicitados y las salidas proporcionadas. Cada proceso que se realice constituirá un aporte de conocimiento compartido que servirá como retroalimentación de jefes a colaboradores como de colaboradores a jefes. Para ello se deben concretar revisiones periódicas de los avances obtenidos y manejar una plataforma de gestión de proyectos que permita ver el estado real del desarrollo de cada componente.
- **Realización de pruebas unitarias al componente.** La realización de pruebas unitarias en cada componente de software deberá estar a cargo de cada colaborador y deben ser ejecutadas independientemente y sin importar el entorno de programación que se esté utilizando.
- **Entrega final del componente.** La entrega final del componente que hará el colaborador deberá ser entregada a su respectivo jefe de zona quien se encargará del acoplamiento general del proceso. Para dichas entregas se hará uso de las tecnologías de la información y comunicación. Por ejemplo, para almacenamiento de código fuente se propone el uso de repositorios lo que permitirá mantener actualizado los cambios que se realicen y se pueda evidenciar la madurez en que el proyecto se encuentra. Para seguimiento de actividades, las zonas deben contar con una herramienta que les ayude a gestionar el rendimiento de cada participante, estimar sus avances y determinar claramente la evolución del proyecto total y finalmente para la comunicación entre colaboradores se recomienda emplear los chats directos, video llamadas grupales y para asumir algún asunto de mayor delicadeza o formalidad el uso de correo electrónico.
- **Entrega de la documentación del componente.** El colaborador deberá entregar al jefe de zona además de un componente funcional, la documentación respectiva que lo relaciona en base al estándar proporcionado.
- **Integración de componentes.** Cada colaborador hará entrega al jefe de zona del componente de software asignado, con su respectiva documentación y una

vez se encuentre validado por las pruebas unitarias realizadas. Dicho componente deberá cumplir el estándar de codificación y el estándar para la gestión y administración de código fuente inicialmente establecido por el líder del proyecto para evitar que exista heterogeneidad en el mismo, así como también en base a la plantilla de definición de componentes que se le presentó.

El jefe de zona será el encargado de acoplar todos los componentes desarrollados por sus colaboradores encargándose además de la codificación resultante para enlazar cada componente uno a otro en caso de que estos tuviesen una interdependencia. Los parámetros de entradas, salidas e interrelaciones que dicha plantilla especifica son las pautas claves para propiciar dicha integración.
Al concluir con el objetivo, el conjunto de componentes formará el proceso que fue asignado a la zona y una vez este se encuentre consolidado, el jefe de zona lo entregará al líder el cual con ello obtendrá el porcentaje de avance del proyecto total.
Para llevar a cabo este proceso de la mejor manera es en la etapa de diseño en donde la especificación de puntos de integración debe estar establecidos.

En la tabla 6 se relacionan de manera resumida las actividades que anteriormente se mencionaron, con sus respectivos roles y los insumos necesarios para su realización.

Tabla 6. Definición de Actividades en base a roles

Roles	Actividad	Descripción Actividad	Insumos	Descripción Insumos
Líder del proyecto	Establecer las funcionalidades, procesos y componentes del sistema.	Se consolidará el proyecto en base a los requisitos recolectados teniendo en cuenta dichos criterios para la división del trabajo colaborativo.	Plantilla de Definición de Funcionalidades	Plantilla que contempla las funcionalidades del sistema.
			Plantilla de Definición de procesos	Plantilla que especifica cada proceso existente en base a las funcionalidades establecidas.
			Plantilla de Definición de Componentes	
	Detallar los parámetros de entrada y salida de cada componente y sus interrelaciones.	Se especificarán que variables recibe cada componente y que resultados debe emitir, así como las posibles conexiones con otros elementos.	Plantilla Definición de Componentes	Plantilla que especifica el componente a desarrollar y los parámetros de entrada y salida que cada componente recibe y envía.
	Especificar el modelado y diseño de cada componente.	Aquí se relacionará los respectivos diseños, diagramas o bosquejos del componente para que el colaborador pueda comprender lo que este debe hacer.	Plantilla Definición de Componentes	Esta plantilla se complementa con la información que relaciona los modelos o diseños propios del componente.

Responsable	Actividad	Descripción Actividad	Insumos	Descripción Insumos
Líder del proyecto	Establecer los lenguajes de programación y plataformas de desarrollo a emplear.	Se especifica la arquitectura del sistema, las plataformas, tecnologías y lenguajes a emplear dentro del desarrollo.	Estándar para plataformas de modelado, entornos de programación y gestión de código fuente.	Documento que detalla que herramientas y lenguajes de programación se van a emplear dentro del proyecto.
	Especificar la forma de presentación del código fuente.	Pretende establecer uniformidad al momento de codificar.	Estándar de Codificación	Documento que da las pautas necesarias para propiciar la organización en el código fuente.
	Especificar el estándar de documentación a seguir durante el desarrollo.	Se unifica la forma de presentación de la documentación que se llevara a cabo por cada componente que se realice.	Estándar para la documentación.	Documento que especifica cómo se debe llevar a cabo la documentación del componente desarrollado.
	Distribuir los procesos a las zonas consolidadas.	Una vez formada las zonas se asignará a cada una, un proceso para ser desarrollado.	Plantilla Información de Zona	Esta plantilla contiene la información de las zonas consolidadas.
			Plantilla Definición de procesos.	Esta plantilla se complementa con la información de la zona a la cual fue asignada el proceso para su desarrollo.

Responsable	Actividad	Descripción Actividad	Insumos	Descripción Insumos
Jefe de Zona	Entregar la descripción de los componentes a los colaboradores de la zona.	Se asignará un componente de software a un colaborador de acuerdo a las habilidades que este tenga.	Plantilla Definición de Componentes	Plantilla que especifica el componente a desarrollar y los parámetros de entrada y salida que cada componente recibe y envía.
	Integrar los componentes	De acuerdo a los parámetros establecidos de entrada y salida se acoplan los componentes los cuales consolidaran el proceso asignado.	Proceso de software	Concluido el acoplamiento de los componentes en base a los criterios establecidos se unifica el proceso.
Colaborador	Revisar la descripción del componente.	La idea es que el colaborador se familiarice con el componente de software que va a desarrollar y lo conozca claramente a través de la plantilla proporcionada y el modelado propuesto.	Plantilla Definición de componente	El componente es asociado a unas características solicitadas y listo para ser desarrollado.
	Codificar el componente.	El colaborador empieza a codificar el componente en base a lo que fue especificado.	Líneas de código fuente.	Aplicación de conocimiento que consiste en transformar los algoritmos, modelado y diseño a un lenguaje de programación.

Responsable	Actividad	Descripción Actividad	Insumos	Descripción Insumos
Colaborador	Realizar pruebas unitarias.	Cada desarrollador realiza sus pruebas unitarias para validar su trabajo.	Escenarios de pruebas unitarias.	Permite tener componentes validados a través de las pruebas unitarias que cada colaborador realiza.
	Entregar parcialmente el componente de software	Entregar software funcional en base a las indicaciones expuestas por el jefe de zona y al estándar de trabajo establecido.	Trabajo colaborativo Con el uso de las TIC Estándar de herramientas de gestión de trabajo	Trabajar para que el componente quede totalmente desarrollado con los criterios especificados cumplidos.
	Reutilizar código fuente.	Empleo de líneas de código reutilizables para la realización de diversas tareas.		
	Entregar la versión final del componente	Tras el desarrollo de cada componente se efectúa su entrega al jefe de zona.		
	Entregar de la documentación del componente.	Con el desarrollo de cada componente se entrega adicionalmente la documentación respectiva a este.	Estándar para la documentación	Descripción completa para el uso del componente.

Fuente. Elaboración propia del autor basado en Custom Development Method. [14]

2.4.2 Recursos de trabajo

Dentro del trabajo distribuido tanto para la organización de la información y el acoplamiento de cada proceso que se realice en del equipo, es importante que se formalice todo esto a través de formatos o plantillas que contribuyan a que exista homogeneidad de cada actividad que se haga y no se generen inconvenientes por falta de comprensión.

Plantillas de Apoyo a la fase de Desarrollo

- **Plantilla Funcionalidades.** Plantilla que contiene todas las funcionalidades o características generales del sistema las cuales buscan ser resueltas con la aplicación de conocimiento en desarrollo de software. Esta plantilla arrojará una descripción, características los procesos que puede involucrar y datos complementarios de la funcionalidad que permitan mostrarla de manera detallada. Las funcionalidades deben ser claramente definidas por el líder del equipo y expuestas al resto del equipo.

Tabla 7. Plantilla Funcionalidades.

Funcionalidades	
Nombre del sistema:	Nombre o identificador del sistema software.
Descripción:	Describe de manera general las funcionalidades del sistema.
Características:	Especifica características del sistema.
Información Adicional:	Información complementaria que contribuya al establecimiento de las funcionalidades.
Número de Procesos:	Número de procesos que involucran.
Fecha de Creación:	Fecha en que se define la funcionalidad.
Observaciones:	Comentarios o posibles notas a tener en cuenta.
Definida por:	Nombre del responsable que estableció la funcionalidad del sistema.

Fuente. Elaboración propia del autor basado en [15]. Tabla Campos propuestos aplicables a todas las actividades.

- **Plantilla Procesos.**

Esta plantilla se encarga de definir los procesos que las funcionalidades del sistema involucran los cuales serán asignados a una zona para que esta se encargue con sus colaboradores al desarrollo del mismo.

Tabla 8. Plantilla Definición de Procesos

Procesos	
Id:	Número de identificación del proceso.
Nombre:	Nombre del proceso
Descripción:	Describe de manera general un proceso.
Número de Componentes:	Número de componentes que el proceso involucra.
Nivel de Prioridad:	Definir qué tan relevante es el proceso dentro de la funcionalidad (Alta – Media – Baja).
Jefe de Zona Asignado:	Nombre del jefe de zona al que es asignado.
Zona Asignada:	Identificador de la zona
Fecha de Creación:	Fecha en que se define el proceso
Fecha de Ejecución:	Fecha de inicio en la realización del proceso.
Duración Estimada.	Tiempo estimado para la realización del proceso.
Observaciones:	Comentarios o posibles notas a tener en cuenta.
Definido por:	Nombre del responsable que definió el proceso.

Fuente. Elaboración propia del autor basado en [15], Tabla Campos propuestos aplicables a todas las actividades.

- **Plantilla Componente.** Esta plantilla contendrá dos secciones importantes. La primera parte está encargada de la información general del componente a desarrollar y la cual a su vez hará parte de un proceso del sistema, cuya información será suministrada por el líder del proyecto. La segunda parte definida ya por el jefe de zona el cual complementará el resto de datos

resaltando características más específicas como son las variables de entrada que recibirá y los parámetros de salida que este arrojará, así como las posibles interrelaciones con otros componentes facilitando que el acoplamiento sea el pertinente y de la mejor manera posible. Aquí ya se ya se tendrá el nombre del responsable encargado y el tiempo estimado para su desarrollo. Cada componente además debería ir ligado a un esquema de modelado y otro de diseño para facilitar su comprensión.

Tabla 9. Plantilla Definición del Componente

Componentes	
Id:	Número de identificación del componente
Nombre:	Nombre del componente.
Descripción:	Describe de manera general un componente.
Características:	Define las características principales del componente.
Entradas:	Insumos de entrada para la realización del componente. Parámetros que recibe.
Salidas Esperadas:	Descripción de parámetros de salida esperados o variables de retorno.
Diccionario de datos:	- Especifica detalles generales de lo que hace cada función o procedimiento si los hubiese dentro del componente. - Establece la terminología que se manejará dentro del componente, evitando así ambigüedad respecto a los nombres a usar, descripciones, contenido. -Detalla flujos de información mucha más precisos sobre el componente a realizar. (Detalles elementales e importantes a considerar, atributos, variables).
Interrelaciones:	Especifica si el componente tiene alguna relación con otro componente y de qué manera se acoplan.
Conexiones:	Detalla las conexiones con gestores o servidores de base de datos.
Diseño del Componente:	Especifica si hay un prototipo guía para el diseño del componente. (Screenshots – Modelado - Diagramas)
Designado a:	Nombre del integrante al cual se le asigna el componente.
Rol que cumple:	Papel que desempeña la persona asignada en la realización del componente.

Zona Responsable:	Identificador de la Zona responsable del componente.
Proceso:	Nombre del proceso al cual pertenece.
Nivel de Prioridad:	Definir la prioridad del componente (Alta – Media – Baja).
Tiempo Estimado:	Aproximación de tiempo en la realización de dicho componente.
Fecha de Inicio:	Fecha de inicio de actividades
Fecha de finalización:	Fecha estimada de finalización de actividades.
Observaciones:	Anotaciones adicionales.
Contrariedades:	Posibles elementos a tener en cuenta dentro del componente
Definido por:	Nombre del responsable que definió el componente.

Fuente. Elaboración propia del autor basado en [15], Tabla Campos propuestos aplicables a todas las actividades.

- **Plantilla de Información de Zona.** Esta plantilla expone información sobre las zonas que fueron definidas para el desarrollo del software. Reúne todos los detalles en cuanto al número posibles de participantes que la conforman, su respectiva ubicación y posible diferencia horaria existente entre ellos.

Tabla 10. Plantilla Información de Zona

ZONA	
Id:	Número de identificación de la zona. Use tres letras que identifiquen el lugar de trabajo más el número de participantes que lo integran.
Descripción:	Información de la zona.
Jefe de Zona:	Responsable de la zona.
Número de Participantes:	Número de colaboradores que conforman la zona.
Presenta Desfase horario:	Informa si dentro de los colaboradores hay un desfase horario (Si – No)
Horas de diferencia:	Número de horas de desfase (Si no hay desfase No Aplica NA)
Colaboradores:	
Nombre	**Ubicación Geográfica**
Define el nombre de cada participante	Lugar donde se reside el colaborador

Fuente: Elaboración propia del autor

- **Estándar de codificación.** Este tipo de documento permitirá llevar a cabo buenas prácticas al momento de codificar, permitiendo que el código fuente se presente de una manera uniforme, se entienda más fácilmente, sea accesible a cambios de manera rápida y el programador siga unas reglas que garanticen el orden y la homogeneidad de cada componente de software que realice. A continuación, se plantea un estándar de codificación que puede ser aplicable en entornos distribuidos y adaptable a cualquier lenguaje de programación que el equipo utilice.

Tabla 11. Estándar de prácticas de codificación

Estándar de *prácticas de* codificación	
Recomendaciones Generales.	Describe brevemente algunas sugerencias al momento de codificar que deben ser tomadas en cuenta por los colaboradores para propiciar la homogeneidad en el código fuente. - Escribir cada componente de software en inglés incluyendo comentarios. - Usar paquetes o diccionarios para mostrar la información al usuario final en el lenguaje nativo. - Las líneas en blanco o comentarios no cuentan como líneas de código. - Separar en archivos la parte de lógica de negocios con la capa de presentación.
Formato de cabecera	Describe como debe organizar la información de cabecera en cada archivo que se realice con el fin de que cada colaborador lo tenga en cuenta al momento de codificar. *@nombre:* Nombre del componente. *@objetivo:* Detalla brevemente el objetivo general del componente. *@descripción*. Detalla brevemente lo que hace el componente. *@colaborador:* Nombre del colaborador que realizó el componente. *@zona:* Identificador de la zona a la cual pertenece. *@fecha:* Fecha de entrega del componente. *@versión:* Número de entrega correspondiente al componente.

<table>
<tr><td>Ejemplo Formato de cabecera.</td><td>

```
@nombre: Componente para el registro de usuarios a la plataforma.
@objetivo: Desarrollar un componente apto para facilitar el registro de usuarios al sistema de notas de la Universidad de Nariño.
@descripción: Componente para el registro de usuarios al sistema sea docentes tiempo completo u hora catedra de la Universidad de Nariño, solicitando la información básica del usuario.
@colaborador: Nelson Suarez
@zona: COME-03
@fecha: 10 – febrero 2017
@versión. 01
```

</td></tr>
<tr><td rowspan="7">Organización de Variables.</td><td>Especifica la manera en cómo las variables que puedan ser usadas, debe ser declaradas siguiendo un orden en específico y además línea por línea.</td></tr>
<tr><td>Ejemplo:
Correcto:

```
int contador = 1;
string saludo = “bye”;

Incorrecto:
string saludo = “bye”; int contador = 1;
```

</td></tr>
<tr><td>- Organización por variables siguiendo el orden de tipo numérica (<code>int, float, double</code>), tipo carácter, tipo texto, tipo booleano, arreglos o matrices.</td></tr>
<tr><td>Ejemplo:
Correcto:

```
int contador = 0;
string nombre = “Maria”;
boolean bandera = true;
string idEstudiante = ‘212’;

Incorrecto:
string saludo = “hola”;
int numero = 2;
```

</td></tr>
<tr><td>- Nombre de variables compuesta con el formato CamelCase.</td></tr>
<tr><td>Ejemplo:
Correcto de la Forma LowerCamelCase:

```
int contarLlegadas = 0;
```

</td></tr>
</table>

Organización de Variables.	**Incorrecto:** `int countllegadas = 0;`
	-No permita que el nombre de las variables sea la combinación de más de dos idiomas.

	Ejemplo:
	Correcto:
	`int contarSalidas = 0;`
	Incorrecto:
	`int countSalidas = 0;`

Inicialización de Variables	Procure inicializar las variables que va a usar al momento de su declaración.
	Ejemplo:
	Correcto
	`int dinero = 0;`
	`String [] profesion = {"Sistemas", “Administración", "Psicología”};`

Constantes	- Se sugiere que el nombre de las constantes se escriba en mayúscula.
	Ejemplo:
	Correcto:
	`int AREA_MAXIMA;`
	`int EDAD_MINIMA;`

Indentación	Propiciar que el código este ordenado en base a las estructuras que se vayan planteando. Se propone una Indentación común de 4 espacios equivalente al uso del tabulador.

Operadores	Permite establecer la declaración de operadores dentro del código fuente.
	Agregar espacios entre:
	- Operandos de declaración de variables.
	Ejemplo:
	`int edad = 5;`
	
	- Entre valores de operaciones aritméticas
	Ejemplo:
	`m = 5 + 2;`
	
	- Entre operadores lógicos o de comparación como `<=, >=, <, >, !=, &&, ||.`

Operadores	Ejemplo:
	`if(var > 4 && var < 10){`
	`  cout<<“Correcto”;`
	`}`
	
	Use signos de agrupación para separar los operadores lógicos cuando se usan más de uno en una misma instrucción.
	
	Ejemplo:
	`if((edad > 15 && edad < 20) || (genero == ‘F’)){`
	`  alert(“comprobado”);`
	`}`

Comentarios	- Los comentarios pueden ser de una línea y de múltiples líneas. - Después de un comentario aplique un salto de línea. - Los comentarios deben ir con lo estrictamente necesario. - Los comentarios deben mantener que no ocupen más de 80 caracteres por línea.

Funciones, clases o métodos internos	-Definir las funciones internas que pueda contener el componente software con nombres pertinentes a lo que se está realizando y que no superen los 30 caracteres.
	
	-Evitar colocar espacios entre el nombre de un método o función y el paréntesis de apertura de esta.
	Ejemplo:
	Correcto:
	`function verEstudiantes()`
	
	- Definir cada método o función empleada con el formato lowerCamelCase.
	Ejemplo:
	Correcto:
	`void consultarRegistro()`
	
	- Coloque espacios en un método o función entre cada coma que separa un parámetro de otro.
	Ejemplo:
	`void consultarNotas(a, b, c);`

Funciones, clases o métodos internos	- Separe cada método, función o clase con una línea en blanco.
	
	- Establecer un conjunto de palabras claves que puedan contribuir a que las funciones internas sean nombradas bajo un estándar similar. Apoyarse en el Diccionario de Datos de cada componente.
	Ejemplos:
	`registrarEstudiante()`
	`registrarDocente()`
	`registrarEmpleado()`
	`visualizarDocente()`

División de líneas.	Cuando una expresión no alcance por su tamaño en una sola línea esta podrá continuar en la siguiente siempre y cuando:
	
	- Exista una coma
	- Exista un operador.
	
	Ejemplo:
	`int recibirCliente(nombre, apellido, edad, teléfono);`
	
	`if ((condicion1 && condicion2)`
	`|| (condicion3 && condicion4) {`
	
	`recibirCliente('maria', 'gomez', 31, '722881');`
	`}`

Reutilización de código	Para evitar que exista código repetido dentro del desarrollo del componente, haga uso de la reutilización de código fuente mediante el uso de funciones, economice tiempo y trabajo doble.

Fuente. Elaboración propia del autor basado en Estándar de Buenas prácticas de codificación. [60] [61] [62] [63] [64]

- **Estándar para plataformas de modelado, entornos de programación y gestión de código fuente.** Establece las características a considerar al momento de desarrollar el componente junto herramientas a emplearse las cuales serán vitales para la realización del trabajo y la administración de código fuente.

Tabla 12. Estándar para plataformas de modelado, entornos de programación y gestión de código

Plantilla de entornos de programación	
Recomendaciones Generales.	Describe brevemente algunas sugerencias al momento de codificar que deben ser tomadas en cuenta por los colaboradores para propiciar que el uso de las plataformas de programación que vayan a emplear sea la misma para todos.
Lenguajes de programación admitidos.	Describe según la preferencia de los equipos los lenguajes de programación que se vaya a emplear para la realización del software. Ejemplo: `Php, Javascript`
Frameworks para su uso.	Especifica los marcos de trabajo que se vayan a usar dentro del desarrollo de software. Ejemplo: `Angular Js, MDBootstrap`
Entornos de desarrollo	Sugiere un de IDE apropiado para desarrollar el componente software. Ejemplo: `Netbeans, Eclipse.`
Editor de texto	Recomienda un editor de texto para trabajar el código fuente. Ejemplo: `Sublime Text, Atom`
Gestor de base de datos	Define la herramienta que va ser empleada para la creación, gestión y administración de las bases de datos dentro del sistema. Ejemplo: `PostgreSQL, MySQL.`
Gestor de librerías o dependencias.	Determina la herramienta que va ser empleada para la administración y gestión del código fuente, como lo es mediante el uso de repositorios. Ejemplo: `Github, Bitbucket`

Modelador de Software	Describe las herramientas útiles para modelar y diseñar software (diagramas UML, diagramas entidad-relación). Ejemplo: StarUML

Fuente: Elaboración propia del autor.

- **Estándar de herramientas de gestión de trabajo.** Es un estándar que se encarga de postular las herramientas que van a servir para el desarrollo de las actividades, tener un seguimiento de estas, y poder trabajar de manera colaborativa todos los integrantes del equipo.

Tabla 13. Estándar de herramientas de trabajo

Plantilla de herramientas de gestión de trabajo	
Recomendaciones Generales.	Describe brevemente algunas sugerencias al momento de usar las herramientas que serán empleadas por los colaboradores, jefes de zona y líder del proyecto.
Plataformas para la comunicación.	
Descripción	Detalla el uso de las herramientas de comunicación que van a ser empleadas dentro del proyecto como parte fundamental para la interacción entre los participantes.
Ejemplo Plataformas para la comunicación.	**Skype.** Empleada para la realización de reuniones grupales de manera virtual cara a cara, entre colaboradores y jefes de zona **Slack.** Empleada para conversatorios individuales o grupales dentro del equipo mediante canales de comunicación. **Correo electrónico**. De carácter formal para él envió de las plantillas de funcionalidades, procesos y componente entre los participantes, así como la asignación de actividades, tutoriales y estándares establecidos.
Tutorial	Indica si anexa tutorial de las herramientas de comunicación a usar. Dicha información se entregará como parte del estándar de documentación.

Plataforma para la administración de proyectos.	
Descripción	Orientada al seguimiento de actividades en línea, de cada una de las tareas que cada colaborador esta efectuado y el estado real de ellas.
Ejemplo para plataforma de administración del proyecto.	**Trello.** De uso diario para la administrar el flujo del proyecto en su totalidad y de los procesos asignados por zona para tener un control de los avances con mayor facilidad, organizado y midiendo su progreso.
Tutorial	Indica si anexa tutorial de las herramientas de administración del proyecto a usar. Dicha información se entregará como parte del estándar de documentación.

Fuente: Elaboración propia del autor.

- **Estándar para la documentación.** Es un estándar que facilita información adicional sobre el sistema, detalles para documentar los componentes desarrollados, glosario de herramientas, tutoriales, pautas para manuales de usuario, etc.

Tabla 14. Estándar para la documentación

Plantilla para la documentación del Sistema	
Patrones de Arquitectura	Detalla la arquitectura del sistema. Ejemplo: Arquitectura Cliente - Servidor, Arquitectura web, Arquitectura móvil.
Fuente	Define parámetros importantes de escritura del sistema tales como: - Tipo de Fuente, tamaño de fuente, estilo, color.
Modelado	Determina la forma de presentación y de detalle de diagramas o diseños propios del componente. Ejemplo: Cada diagrama, modelado o diseño que se haga del componente deberá tener un título y si se desea una descripción sencilla (no supere las 20 palabras).

Interfaces	Detalla las especificaciones complementarias de las interfaces que se realicen.
	- Efectos, Diseños, Colores.
	- Gráficos.
	- Elementos Audiovisuales.
	
	Ejemplo:
	`Color de Interfaces para fondo:`
	`HEX #E88080 o RGB 232, 128, 128`
	`Color de Interfaces para contraste:`
	`HEX #F6D9D9 o RGB 246, 217, 217`

Glosario de Herramientas	-Terminología de Herramientas. Describe brevemente sobre las herramientas que se van a usar si el desarrollador los desconoce.
Tutorial de Herramientas.	Anexa tutorial de instalación y manejo de herramientas que se vayan a usar dentro del desarrollo del software.
Manual de usuario.	Detalla la manera de escribir el manual de usuario para el aplicativo.

Fuente. Elaboración propia del autor basado en [65]

2.5 Definición de la práctica para equipos sin restricciones de tiempo y espacio de acuerdo a los principios de SEMAT.

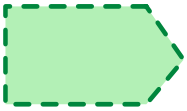
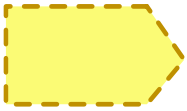
Durante toda la investigación se hizo una profundización sobre el desarrollo de software en equipos distribuidos y justamente en la etapa anterior se determinaron las actividades, roles y herramientas que caracterizaron la práctica. Finalmente, para cumplir con el objetivo final es necesario representar la práctica con base a los principios de SEMAT.

2.5.1 Introducción a la representación gráfica de SEMAT

Una vez conocido el núcleo de SEMAT, así como los elementos que agrupa (alfas, espacio de actividad y competencias) se procede a efectuar la representación de la práctica de forma visual de acuerdo a la sintaxis gráfica del lenguaje.

Los elementos gráficos del lenguaje a considerar son:

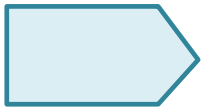
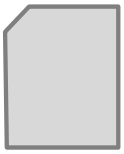
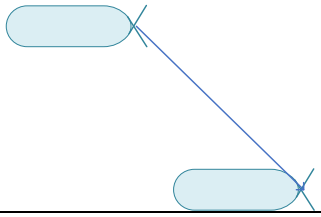
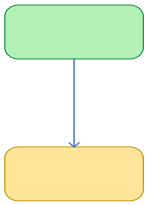
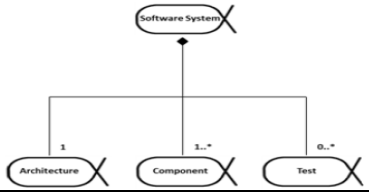
Tabla 15. Aspectos del núcleo de SEMAT.

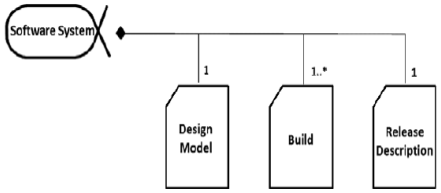
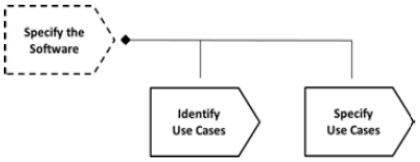
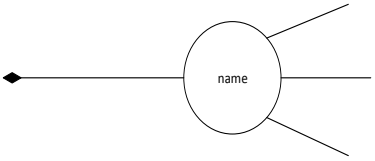
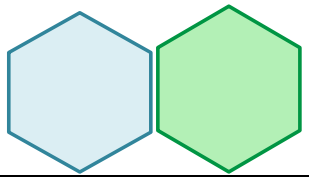
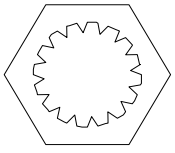
Elemento	**Descripción**	**Representación gráfica.**		
		Cliente	**Solución**	**Esfuerzo**
Alfa	Cosas esenciales para trabajar			
Espacio de actividad.	Conjunto de actividades que siempre se hacen.			
Competencia	Capacidad para realizar una actividad.			

Fuente. Elaboración propia del autor basado en [13] [14]

Para la asociación de dichos elementos se tiene:

Tabla 16. Sintaxis grafica de los elementos del núcleo.

Elemento	Descripción	Representación gráfica.
Actividad	Define una o más clases de productos de trabajo y brinda una guía sobre cómo se usan estos elementos cuando se utiliza una práctica.	
Producto de Trabajo	Artefacto de gran valor obtenido tras un esfuerzo de ingeniería de software.	 Work product
Patrón	Es una descripción de una estructura en una práctica. Se relaciona con el rol que desempeña la actividad.	 Pattern
Asociación de alfas	Se define como una línea continua que sirve para conectar dos alfas.	
Sucesor de estados	"Se define como una línea sólida con una flecha abierta que conecta un estado con su sucesor."	
Alpha Containment	"Es una línea continua vertical para conectar un alfa con un subalfa con un diamante relleno en la punta hacia el lado del alfa."	

Manifiesto de producto de trabajo	“Se visualiza como una línea continua horizontal para conectar un alfa con un producto de trabajo con un diamante relleno en la punta hacia el lado del alfa.”	
Asociación de actividad	“Se visualiza como una línea continua horizontal para conectar un espacio de actividad con una actividad con un diamante relleno en la punta hacia el lado del espacio de actividad.”	
Asociación de patrón.	“Formado por una línea sólida con punta de diamante relleno hacia el lado del patrón, originada desde un círculo que se conecta a su vez con cada uno de los elementos asociados con el patrón. El nombre de la asociación se pone dentro del círculo.”	
Método	“Conjunto de prácticas que relacionan todas las actividades que se involucran en el desarrollo de software.”	
Práctica	“Enfoque que se repite con un objetivo en específico. Un método puede contener varias prácticas. “	
Kernel	“El núcleo se genera a partir de métodos. Los métodos específicos contienen las prácticas y éstos se incluyen en el núcleo.”	

Fuente. Tomado de [13] [14]

2.5.2 Constitución de la práctica con base al núcleo de SEMAT.

SEMAT dentro del campo de Ingeniería de software ha propuesto la apropiación de prácticas permitiendo así estas sean usadas en cualquier circunstancia de acuerdo con las necesidades que se requieran y no necesariamente regirse por un modelo en específico. Siguiendo dicha perspectiva y mediante el estado del arte efectuado, el reconocimiento de roles, actividades y herramientas se propone orientarla con base en el estándar de Essence que la comunidad de SEMAT ha dado a conocer en los últimos años.

a) Roles con base al núcleo de SEMAT

SEMAT aún no contempla la parte de roles en su estándar es por eso que estos se manejaran en base a la formulación que se tenía inicialmente, excepto el rol de Colaborador quien se le da un término mucho más apropiado de acuerdo a las actividades que realiza.

- **Líder del proyecto:** Persona a cargo de la dirección general del proyecto y del equipo de trabajo.
- **Jefe de Zona:** Persona que está a cargo de la zona asignada y vela por la realización de trabajo en su respectivo subequipo.
- **Desarrollador**: Es el Colaborador. Es la persona que contribuye al desarrollo del software.

b) Alfas que involucra la práctica

De acuerdo con el núcleo de SEMAT se define a las alfas como las cosas con las que siempre se trabajan. Permiten profundizar claramente cada esfuerzo de Ingeniería de Software que se realiza, medir que tan bueno fue y estar atento a su evolución. Cabe destacar que para evaluar dicho progreso las alfas cuentan con estados los cuales a su vez poseen listas de chequeo que a medida que cada uno de los ítems se cumplan pasara de un estado a otro. Las alfas se comportan como unos indicadores de monitoreo y avance y están distribuidas en 3 áreas de conocimiento, la del cliente, la de solución y esfuerzo. [13]

Para el caso de la práctica, de las siete alfas que actualmente el estándar propone, se tomaran únicamente las ubicadas en el área de conocimiento solución, puesto que establece la forma en como un equipo define los requisitos, los implementa, construye, testea y mantiene un sistema. Dentro de esa área están el alfa de Sistema Software y el alfa de Requisitos.

La selección de esas alfas fue porque las actividades que involucra son netamente relacionadas a la parte de construcción y codificación del software y claramente se evidencia que estas apuntan a ese objetivo.

De [13] se extraen los estados que involucran el alfa de Requisitos y Sistema Software y se especifican en la siguiente tabla:

Tabla 17. Alfa Requisitos, estados y listas de chequeo

Alfa: Requisitos		
Estado	**Descripción**	**Lista de Chequeo**
Concebido	La necesidad de un nuevo sistema ha sido establecida.	• Los interesados están de acuerdo en se debe producir un sistema. • Se identificaron los usuarios del sistema • Se identificaron las partes que financiaran dicho sistema. • Existe una oportunidad clara para que el sistema se aborde.
Acotado	El propósito y extensión de un nuevo sistema es claro.	• Identificación de partes interesadas en el desarrollo del sistema. • Se acordó el propósito del sistema. • Está claro el acierto del nuevo sistema. • Los interesados tienen en cuenta el alcance del sistema. • La forma de obtener los requisitos es establecida. • Los mecanismos para gestionar los requisitos están en su lugar. • El esquema de priorización es claro. • Las restricciones son identificadas. • Conjeturas claramente establecidas.

Coherente	Los requerimientos proporcionan una descripción consistente de las características esenciales del nuevo sistema.	• Los requisitos son compartidos y asimilados por el equipo y los interesados. • El origen de los requerimientos es claro. • Los conflictos presentados con los requisitos fueron identificados y resueltos. • Se identifican las características esenciales del sistema. • Se identificaron importantes escenarios de casos de uso. • Se priorizan los requisitos. • Se define el impacto de los requisitos. • El equipo asimila lo que debe ser entregado.
Aceptable	Los requerimientos describen un sistema que es aceptable por los interesados.	• Las partes interesadas aceptan que los requisitos describen una solución aceptable. • La Tasa de cambio a los requisitos iniciales es relativamente mínima. • El valor de los requisitos es claro.
Tratado	Suficientes requisitos han sido abordados para un nuevo sistema para satisfacer las necesidades de los interesados de manera aceptable.	• Suficientes requisitos han sido cumplidos. • Los requisitos ya implementados aportan un valor claro. • Los interesados aceptan la implementación de los requerimientos en base a lo que hace y no hace el sistema.
Cumplido	Los requisitos han sido tratados completamente satisfaciendo las necesidades para ese nuevo sistema.	• Los interesados aceptan el sistema. • No hay requisitos pendientes. • Se cumplieron los requisitos en su totalidad.

Fuente. Extraído de [13].

Tabla 18. Alfa Sistema Software, estados y lista de chequeo.

Alfa: Sistema Software		
Estado	**Descripción**	**Lista de Chequeo**
Con Arquitectura Seleccionada	Se ha seleccionado una arquitectura para el sistema que aborda los riesgos técnicos y algunas posibles restricciones.	• Los criterios de la selección en dicha arquitectura han sido acordados. • Plataformas de hardware definidas. • Lenguajes de programación y tecnologías han sido seleccionadas. • El alcance del sistema es conocido. • Se tomaron decisiones sobre la organización del sistema. • Los principales riesgos técnicos son acordados.
Demostrable	Existe una versión ejecutable del sistema para demostrar que se implementó con la arquitectura seleccionada y admite pruebas.	• Principales características de la arquitectura son demostradas. • El sistema puede ejercer la interfaz crítica y las configuraciones del sistema. • Los interesados acordaron que la arquitectura es la adecuada.
Usable	El sistema es usable y demostrable con las características de ser operativo.	• El sistema es usable y con las características deseadas. • Los usuarios pueden operar el sistema. • Se aceptaron niveles de defectos. • El sistema está documentado.
Listo	El sistema fue aceptado por un entorno real.	• Listo para ser instalado • Se puso la documentación del usuario a disposición. • Los representantes de los interesados aceptaron el sistema. • Hay soporte operativo.

Operacional	El sistema funciona ya en un entorno operacional.	• El sistema está puesto a disposición de los interesados. • El sistema es compatible con los niveles de servicio acordados. • Al menos un ejemplo del sistema es completamente operacional
Retirado	El sistema ya no es compatible.	• El sistema ha sido reemplazado o descontinuado. • El sistema ya no es compatible • No se producirán más actualizaciones al sistema.

Fuente. Extraído de [13].

c) Espacios de actividad que relaciona la práctica.

La práctica a parte de los alfas que anteriormente se profundizaron, relaciona espacios de actividad las cuales se definen como el conjunto de actividades que ingeniería de software comúnmente se hacen.

La práctica al abarcar únicamente el área de solución, esta cuenta con seis espacios de actividad los cuales sirven de apoyo en la clasificación de actividades que se visualizaron en la tabla 6.

Los espacios de actividad son: [13]

- **Comprender los requisitos.** Se refiere a establecer claramente lo que sistema hará, delimitando su alcance.
- **Darle forma al sistema.** En esta sección ya se estructura el sistema, se define su arquitectura de tal manera que sea más fácil de codificar, modificar, probar y estar dispuesto a posibles mantenimientos de acuerdo a las necesidades futuras.
- **Implementar el sistema.** Aquí ya se desarrolla el sistema en base a unos criterios de entradas con el fin de obtener un producto funcional, efectuando la corrección de posibles errores de acuerdo las pruebas unitarias implementadas y obtener una posible integración con otros elementos del mismo sistema.
- **Probar el sistema.** En este espacio ya se verifica que lo que pidió el interesado se cumpla en su totalidad, identificando cualquier defecto del sistema.
- **Desplegar el sistema**. Una vez probado el sistema, este pasa a manos de los interesados para que estos lo usen y observen su funcionamiento.
- ***Operar el sistema.*** El sistema ha sido operado por el usuario y se apoya su uso en un entorno real.

Para el caso de la práctica esta tomará hasta el espacio de actividad de implementar el sistema, puesto que solo se profundiza la etapa de codificación en equipos de manera distribuida, por ende, la sección de pruebas haría parte de otra práctica la cual establezca la forma en como estas se harían y se validaría el sistema de manera total.

d) Competencias que la práctica involucra.

Las competencias son esas habilidades que una persona dentro del equipo puede llegar a poseer para realizar una actividad y desempeñarla de la mejor manera. Existen 6 tipos de competencias y estás si importar en el área que se encuentren pueden ser aplicadas en cualquier tipo de circunstancias. Estas son:

- **En el área de Cliente esta**:

  Representación del interesado. Esta competencia relacionada con la capacidad de transmitir las necesidades de los interesados y representarlas de la mejor manera para su posterior comprensión. [13]

- **En el área de Solución están**:

  Análisis. Esta competencia agrupa la capacidad que se tiene de comprender las necesidades de los interesados y transformarlas en un conjunto de requisitos bastante entendibles y coherentes en busca de una solución apropiada y de alto nivel. La persona que posea esta competencia tiene un alto criterio de abstracción, es capaz de indagar las causas de los problemas, detallar claramente cada requerimiento y descomponerlo para su mejor visualización. [13]

  Desarrollo. Esta competencia congrega la capacidad que se tiene para diseñar y programar un sistema software siguiendo las normas y estándares establecidos por el equipo con el fin de obtener algo funcional y que cumpla con el conjunto de requerimientos propuestos por el interesado. Dicha persona con esta competencia tiene habilidades y conocimientos en lenguajes de programación, diseño y un pensamiento crítico en busca de obtener la mejor solución posible al sistema. [13]

  Pruebas. Esta competencia encierra dichas capacidades que se tienen para probar un sistema y verificar que funcione siempre de la mejor manera. La persona encargada de testear el sistema deberá ser capaz de proporcionar las pruebas correctas, evaluar que los requisitos se cumplieron a cabalidad y observar detalladamente cada falencia que pueda este tener para propiciar su corrección de manera inmediata. [13]

- **En el área de esfuerzo se encuentran:**

  Liderazgo. Esta competencia relaciona la facultad que tiene un individuo en liderar un grupo de personas y trabajar con ellas de manera objetiva para el logro de cada una de las metas propuestas. Además, contribuye a propiciar la comunicación, la colaboración, a motivar a su equipo a cumplir con sus actividades y eliminar cualquier tipo de barrera que pudiese impedir la labor. [13]

  Gestión. Esta competencia encapsula la forma de coordinar el trabajo de un equipo y darle seguimiento a este, observando los resultados que se han obtenido. La gestión contribuye a planificar y organizar cada tarea que se asigne a cada participante y estimar de mejor manera los riesgos. [13]

2.5.3 Actividades de la práctica representadas en base al núcleo de SEMAT

Una vez definidos los elementos que involucrará la práctica se prosigue a clasificar cada una de las actividades que inicialmente se mencionaron con base en los espacios de actividad del área solución, con sus respectivas alfas (requisitos, sistema software), el cambio de estado que cada alfa involucra, roles (responsables), recursos empleados (insumos) y los productos de trabajo obtenidos al final de cada una de ellas.

Aunque la práctica no abarca el análisis de requerimientos ni la parte del diseño, el alfa Requisitos está presente porque es ahí donde la necesidad del nuevo sistema ha sido establecida para ser llevada a cabo por un conjunto de desarrolladores y trabajada de manera conjunta para alcanzar el objetivo final que es la implementación del Sistema software.

Tabla 19. Actividades de la práctica en base al Núcleo de SEMAT

Espacio de Actividad	Alfa	Estado	Actividades	Recursos	Producto de Trabajo	Rol
Comprender los requisitos	Requisitos	Coherente	Establecer las funcionalidades, del sistema.	Plantilla de Definición de Funcionalidades.	Documento con Funcionalidades Definidas.	Líder del proyecto
			Establecer los procesos del sistema.	Plantilla de Definición de Procesos.	Documento con Procesos Definidos.	
			Establecer los componentes del sistema.	Plantilla de Definición de Componentes	Documento con Componentes Definidos.	
Darle forma al sistema	Requisitos	Aceptable	Detallar los parámetros de entrada, salida, interrelaciones y dimensión del componente	Plantilla de Definición de Componentes.	Documento con las Especificaciones del componente.	
			Especificar el modelado y diseño de cada componente.		Modelado y Diseño del componente.	
	Sistema Software	Con arquitectura seleccionada	Establecer los lenguajes de programación y plataformas de desarrollo a emplear.	Estándar para plataformas de modelado, entornos de programación.	Documento con Lenguajes y plataformas de programación definidas.	

Darle forma al sistema	Sistema Software	Con arquitectura seleccionada	Especificar la presentación del código fuente.	Estándar de Codificación	Documento con los Parámetros de codificación establecidos.	Líder del proyecto
			Especificar el estándar de documentación.	Estándar para la documentación.	Documento con los Parámetros de documentación definidos.	
			Distribuir a cada zona el proceso correspondiente con sus respectivos componentes.	-Plantilla de Información de Zona -Plantilla de Definición de Procesos.	Zona Consolidada y asignada a un proceso.	
			Entregar la descripción del componente a desarrollar.	-Modelado y Diseño del componente. -Plantilla de Definición de componentes.	Componente asignado a desarrollador.	Jefe de Zona

Implementar el Sistema	Sistema Software	Demostrable	Revisar la descripción del componente.	-Modelado y Diseño del componente. -Plantilla de Definición de componentes.	Componente comprendido	Desarrollador
			Codificar el componente	Lenguajes y plataformas de desarrollo.	Código fuente	
			Reutilizar código fuente.			
			Entregar parcialmente el componente.	Herramientas para la gestión de trabajo	Avance del componente	
			Realizar pruebas unitarias al componente		Escenarios de pruebas unitarias	
		Usable	Entregar la versión final del componente.	Herramientas TIC.	Versión final de componente software	
			Entregar la documentación del componente	Documento con los Parámetros de documentación definidos.	Documentación del componente	
		Listo	Integrar los componentes.	Componentes de software	Proceso formado	Jefe de Zona

Fuente: Elaboración propia del autor

2.5.4 Representación gráfica de la practica en base al núcleo de SEMAT

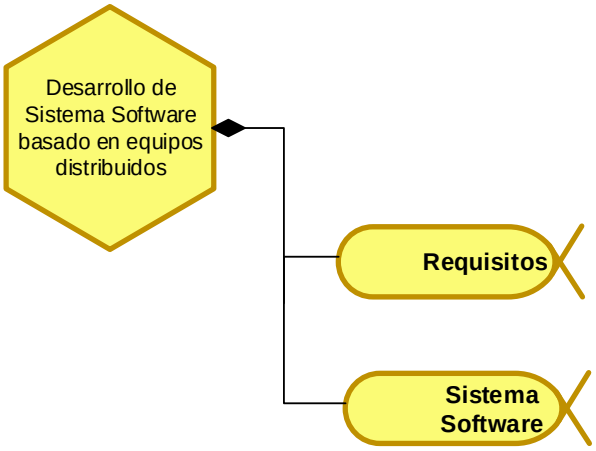
Una vez clasificadas las actividades con sus respectivos elementos, se prosigue a efectuar su representación gráfica por cada espacio de actividad tomando una relación clara entre estos y facilitando su comprensión. Como la práctica está en el área de solución, se representará por un rombo de color amarillo y cuyo nombre se encuentra en el interior de la siguiente figura.

Figura 18. Práctica del área de conocimiento solución.

Fuente. Elaboración propia del autor.

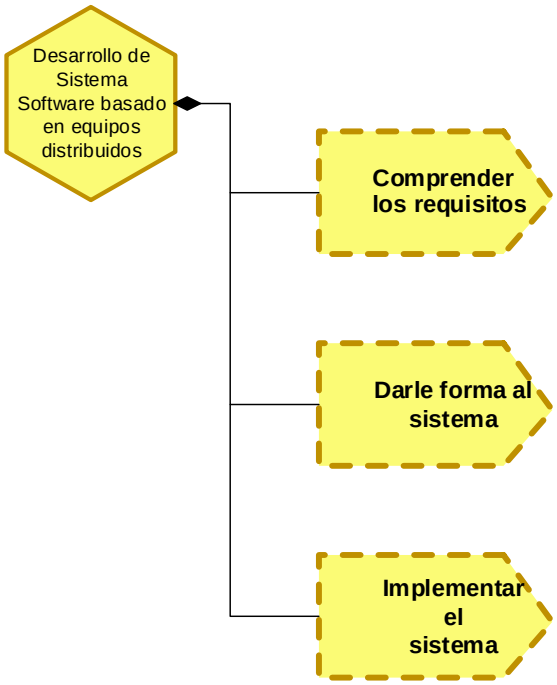
A continuación, se relaciona la práctica con las alfas que involucra y sus respectivos espacios de actividad.

Figura 19. Definición de la práctica mediante los alfas



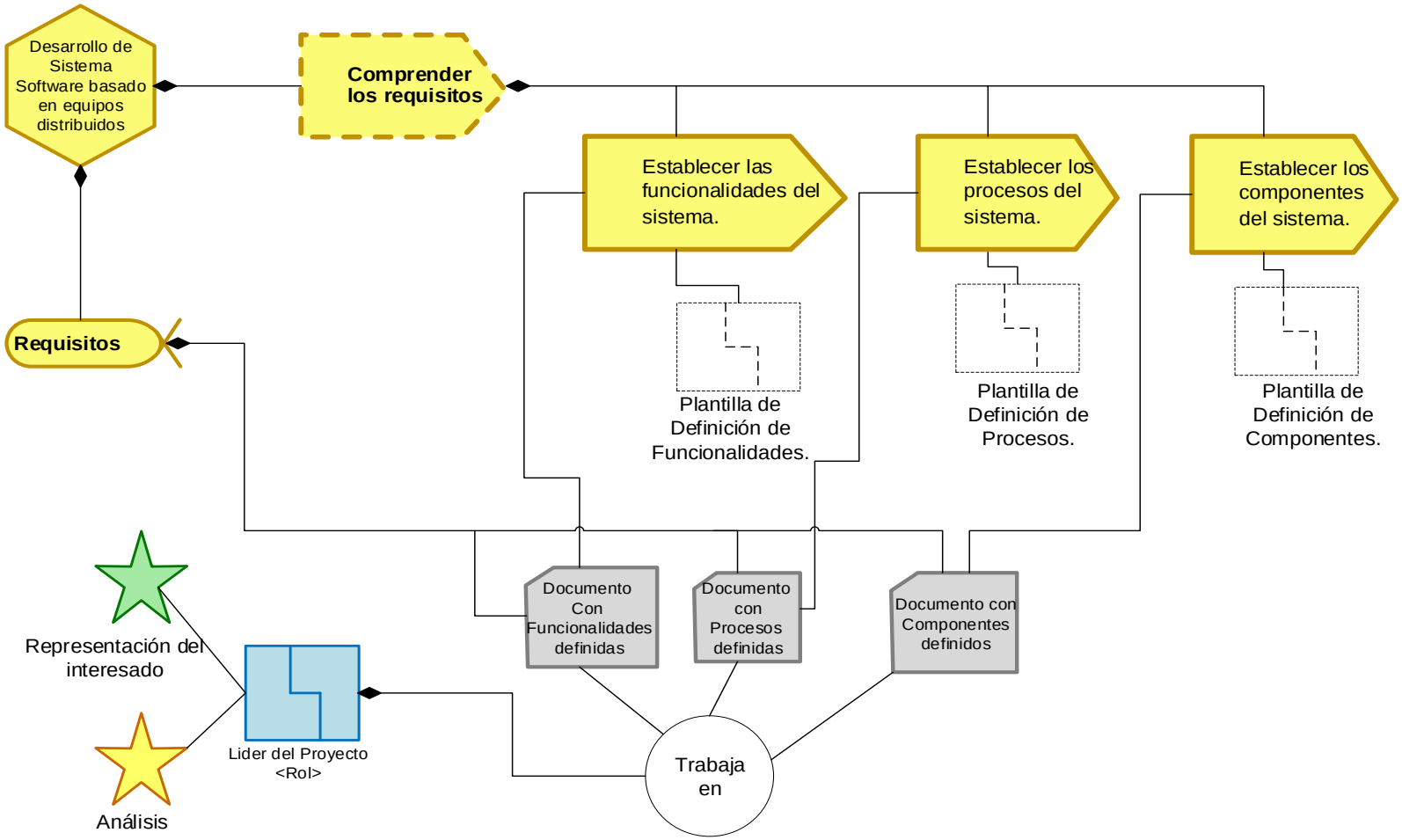
Fuente. Elaboración propia del autor.

Figura 20. Definición de la práctica mediante los espacios de actividad.



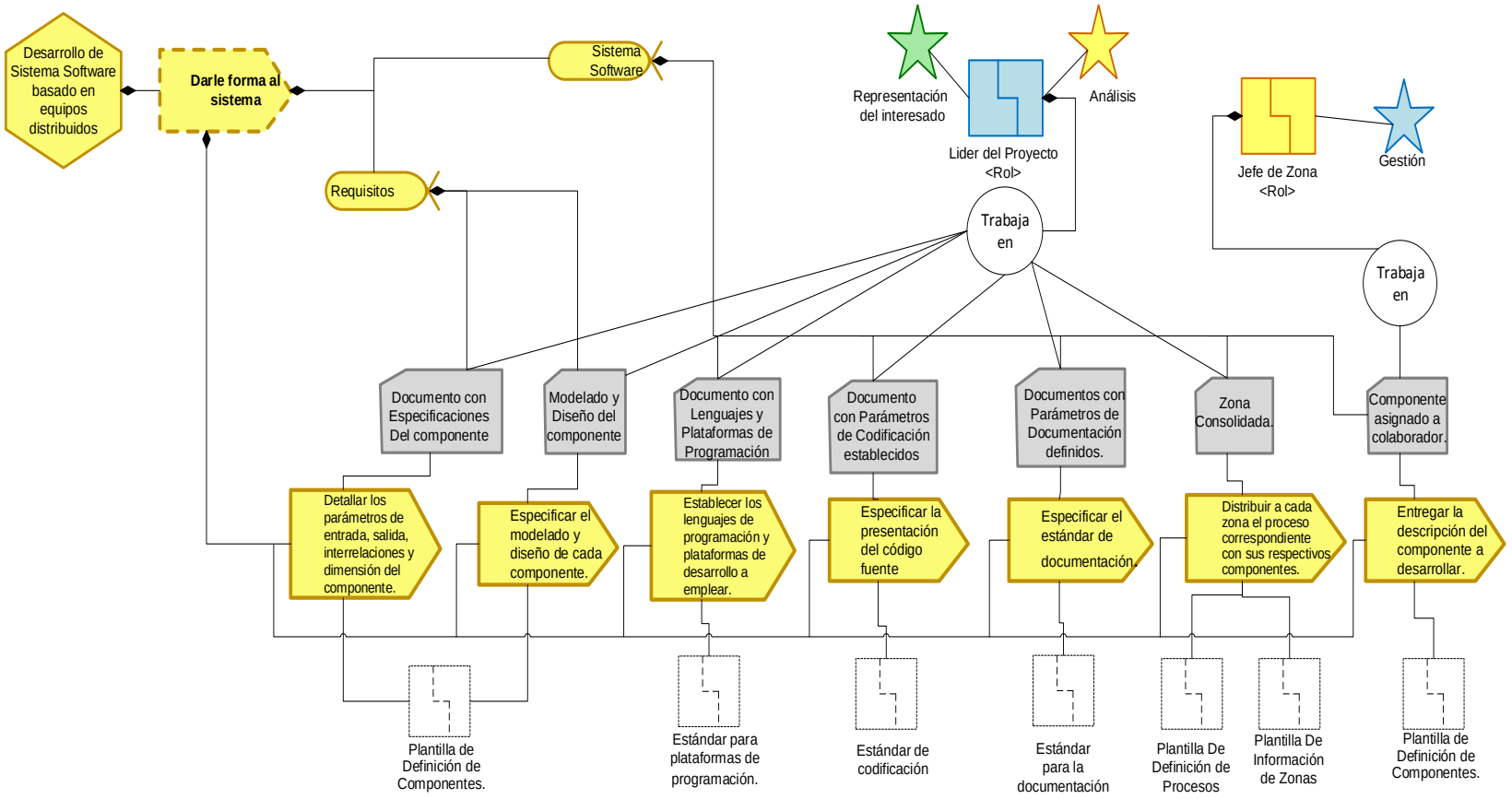
Fuente. Elaboración propia del autor.

Figura 21. Representación gráfica del espacio de actividad ‘Comprender Requisitos’, con actividades, recursos, productos de trabajo, roles y competencias.



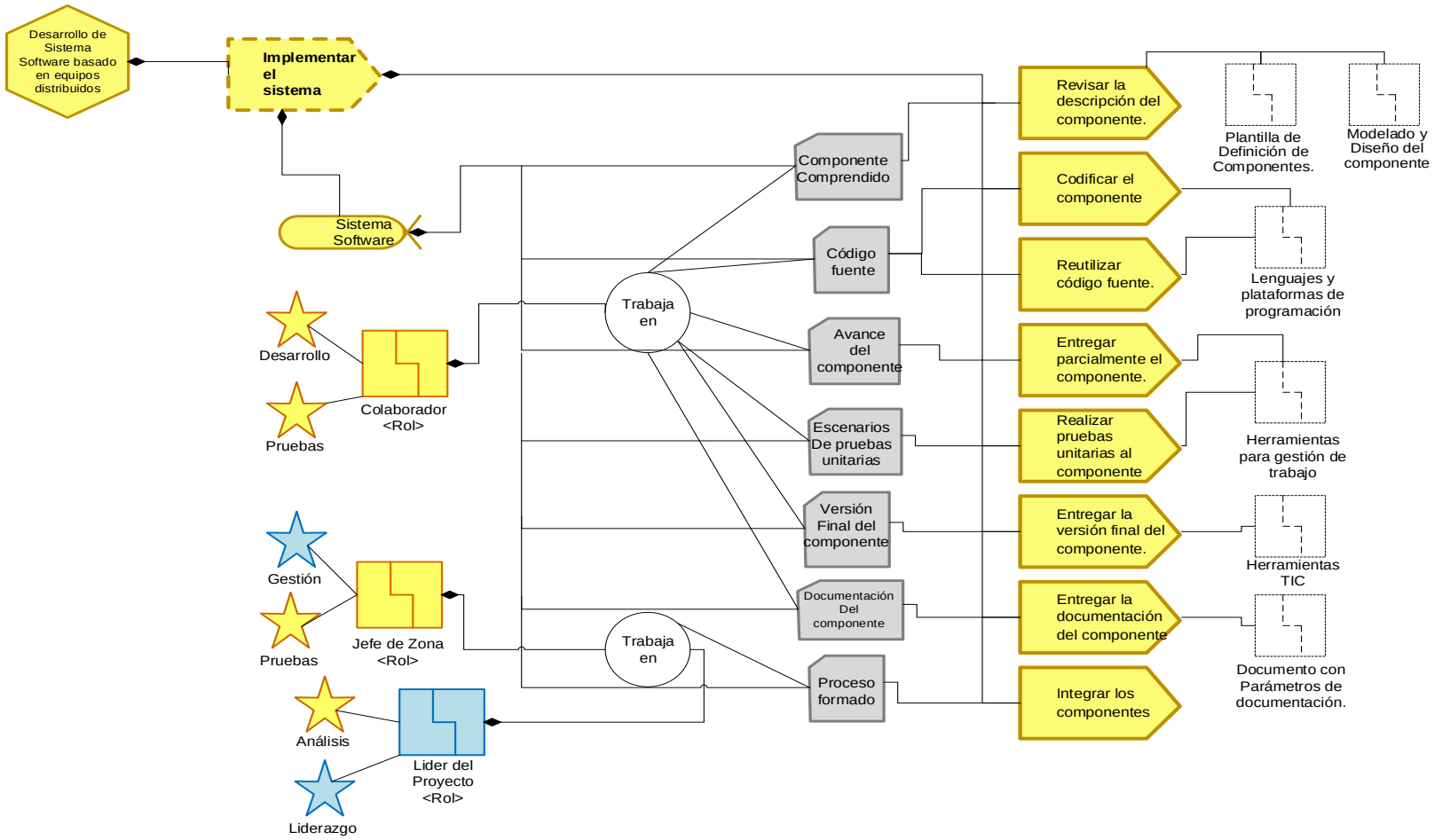
Fuente. Elaboración propia del autor.

Figura 22. Representación gráfica del espacio de actividad ‘Darle forma al sistema’, con actividades, recursos, productos de trabajo, roles y competencias.



Fuente. Elaboración propia del autor.

Figura 23.Representación gráfica del espacio de actividad ‘Implementar el sistema’, con actividades, recursos, productos de trabajo, roles y competencias.



Fuente. Elaboración propia del autor.

2.5.5. Caracterización de la Práctica: "Desarrollo de Sistema Software basado en equipos distribuidos" utilizando el Modelo para la Definición Unificada de la Práctica como Constructo Teórico en Ingeniería de Software.

La Tesis Doctoral del profesor Alexander Barón [66], codirector de la presente investigación propone un modelo que busca Consolidar un modelo para la definición unificada y carente de ambigüedad de la práctica como constructo teórico en ingeniería de software. El modelo soporta una definición unificada porque integra la visión de diferentes propuestas de ingeniería de software. Además, constituye una definición carente de ambigüedad porque permite identificar inequívocamente prácticas de software.

Para caracterizar y nombrar la práctica en Ingeniería de Software, el profesor Barón, en su tesis doctoral, propone una estructura que integra: un verbo nominalizado, que resume el conjunto de actividades que constituyen la práctica; un adjetivo, que indica la forma de abordar la actividad y un objeto sobre el cual se ejecuta la actividad.

El profesor Barón, en su tesis doctoral [66], también propone una taxonomía de verbos, una lista de adjetivos y un conjunto de objetos para nombrar y caracterizar prácticas en ingeniería de software. Estos elementos son el resultado del análisis de prácticas propuestas por diversos enfoques de ingeniería de software, por ejemplo: CMMI, PMBOK, SCRUM, RUP y el conjunto de buenas prácticas de Caper Jones.

Para nombrar y caracterizar la práctica "Desarrollo de Sistema Software basado en equipos distribuidos" se tiene en cuenta que:

- Involucra el área solución del núcleo de SEMAT y a tres espacios de actividad: Comprender los requisitos, Darle forma al sistema e Implementar el sistema.
- Relaciona dos alfas: Requisitos y Sistema Software, que cambian de estado. El alfa Requisitos cambia desde estado Coherente hasta estado Aceptable y el alfa Sistema Software cambia desde estado Arquitectura Seleccionada hasta estado Listo.

De acuerdo con las anteriores consideraciones y utilizando la taxonomía de verbos, la lista de adjetivos y el conjunto de objetos propuestas del profesor Barón en su Tesis Doctoral, se tiene que práctica "Desarrollo de Sistema Software basado en equipos distribuidos" debe tener el siguiente nombre:

El Objeto hace referencia al objeto que más evoluciona como resultado de la práctica, en este caso es ***Sistema Software***.

El Adjetivo se refiere a la forma de abordar la práctica, teniendo en cuenta que la práctica se realiza sin restricciones de tiempo y espacio, el adjetivo es ***Global***.

El verbo nominalizado se define a partir del espacio de actividad que genera la mayor evolución del objeto, en este caso el espacio de actividad es Implementar el Sistema, consultando la taxonomía de verbos encontramos que el verbo nominalizado es ***Implementar.***

En tal sentido, al aplicar el Modelo para la Definición Unificada de la Práctica como Constructo Teórico en Ingeniería de Software, el nombre de nuestra práctica es:

Figura 24. Nombre de la práctica

Fuente. Elaboración propia del autor.

2.5.6 Caso de Estudio: Aplicación de la práctica "Implementación Global de Sistema Software" en un contexto real.

Una vez definida la práctica en base a los principios de SEMAT y de acuerdo a la tesis Doctoral del Ingeniero Alexander Barón, se decidió aplicar la investigación en un contexto real como un complemento adicional y aparte de los objetivos específicos que se plantearon desde un comienzo.

Este caso de estudio se hizo con la finalidad de formular un ejercicio de programación, para ser desarrollado de manera grupal, donde cada integrante trabaje desde su casa o lugar que desee sin necesidad de recurrir a reuniones de forma presencial y aplicar cada una de las actividades de la práctica que durante toda la investigación se dieron a conocer con el propósito de estimar resultados de dicha labor.

Para ello se decidió tomar un ejercicio básico de programación denominado Máquina expendedora del curso APO1 de la Universidad de los Andes [66].

Una vez establecido el problema a trabajar se optó por conformar el equipo de participantes, el cual quedaría consolidado por 4 estudiantes del programa de Ingeniería de Sistemas de la Universidad de Nariño, seleccionados por sus habilidades en el área de programación quien uno de ellos asumiría el rol de jefe de zona y el resto de desarrolladores; todo esto se documentó en la plantilla de Información de Zona.

Posteriormente, se adaptó el ejercicio [66] a los recursos de trabajo: plantillas de definición de funcionalidades, procesos y componentes. Además, se estableció la arquitectura del sistema, el lenguaje de programación a utilizar que para el caso fue Java, los entornos de trabajo (NetBeans), el estándar de codificación y documentación a seguir y las herramientas a utilizar para el seguimiento de actividades y comunicación con el equipo (Trello, Correo electrónico y Skype). Véase Anexos.

El equipo estuvo trabajando en el ejercicio por alrededor de una semana y media y al finalizar se hizo un análisis y estudio con base en los productos obtenidos.

Al culminar las actividades se efectuó una encuesta a los interesados y se les pregunto sobre su punto de vista tras la experiencia de haber aplicado la práctica y al haber usado las plantillas de apoyo.

Entre los aspectos a considerar se resalta:

- La Información proporcionada por el líder del proyecto al equipo se hizo de manera comprensible.

- Forma de trabajo que se llevo a cabo (en cuanto a la asignación de roles, división del proyecto software en base a funcionalidades, procesos y componentes y las herramientas usadas)
- Nivel de claridad proporcionada en las plantillas.
- Roles en el equipo de desarrollo al trabajar de manera distribuida.
- Dificultades presentadas y recomendaciones futuras.

Los resultados se pueden visualizar a continuación.

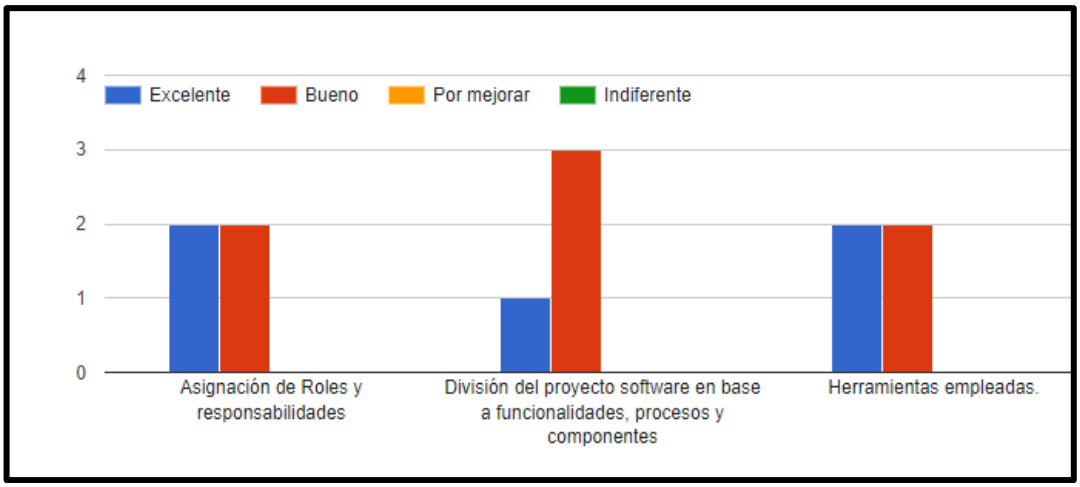
Información proporcionada

En cuanto a la información que se proporcionó a los participantes por parte del líder del proyecto, se puedo observar que su porcentaje de aceptabilidad fue del 100%, considerándolo como un aspecto positivo, puesto que tanto el jefe de zona como los desarrolladores manifestaron que este proceso se llevo a cabo de manera satisfactoria empleando las TIC y gracias a la documentación adjunta que se dio a conocer en el momento oportuno.

Forma de trabajo

De acuerdo a la forma de trabajo que se llevó a cabo se entablo tres aspectos relevantes a considerar: La asignación de roles, la división del proyecto en base a funcionalidades, procesos y componentes y las herramientas empleadas; de los cuales tuvieron un nivel de aceptación favorable, estando en nivel excelente y bueno tal y como se observa en la figura 25.

Figura 25. Resultados Aplicación de la Práctica. Forma de trabajo.



Fuente. Elaboración propia del autor.

Uso de las plantillas.

En esta sección se optó por preguntar a los participantes; qué tan claras fueron las plantillas proporcionadas por el líder del proyecto para el manejo de información. Los resultados se observan en la tabla 20.

Tabla 20. Resultados Aplicación de la Práctica. Uso de Plantillas

Nombre de la plantilla	**Excelente**	**Bueno**	**Por Mejorar**	**Indiferente**
Plantilla de Definición de Funcionalidades	50%	25%	25%	0%
Plantilla de Definición de Procesos	25%	50%	25%	0%
Plantilla de Definición de Componentes	75%	25%	0%	0%
Estándar de Codificación	100%	0%	0%	0%
Estándar de Documentación	25%	75%	0%	0%
Estándar de Herramientas de Trabajo	50%	50%	0%	0%
Estándar de Lenguajes y plataformas de programación.	75%	25%	0%	0%

Fuente. Elaboración propia del autor.

Según los datos arrojados en la encuesta, la mayor parte de las plantillas tuvo un nivel de aceptabilidad Bueno; resaltando que el estándar de codificación fue el que obtuvo una mayor puntuación y se recomienda mejorar y complementar aspectos de la plantilla de definición de funcionalidades y de procesos. Entre las mejoras propuestas por los participantes del ejercicio práctico fue la de agregar a las funcionalidades el ítem de prioridad para que estas sean organizadas de manera descendente de acuerdo a su nivel de importancia, aspecto que podría mejorar la comprensión de estas, la urgencia que se tienen en su realización y lógicamente el objetivo principal del proyecto. También otra de las sugerencias por parte de la plantilla de procesos es la de reemplazar de cierta manera el ítem de duración por uno que se denomine fecha límite para generar un impacto más fuerte al momento de leerse y dar a entender al lector que es algo prioritario y que se tiene hasta cierta fecha para concretar con todo el proceso.

Rol de trabajo

Los resultados obtenidos en base al rol asignado y a la interacción con otros participantes fue buena ya que la distribución de trabajo fue bien realizada, clara y apropiada.

Por parte del jefe de zona este manejo una relación adecuada con el grupo, hizo uso de herramientas de comunicación como Skype para despejar dudas; el correo electrónico para él envío de algún tipo de información adicional y de Trello para el seguimiento de actividades.

Integración de componentes.

En el ejercicio práctico se evidenció la responsabilidad que tenía el jefe de Zona sobre la asignación de componentes y la integración de estos para consolidar el proyecto final, en ese sentido los resultados fueron realmente buenos ya que la información proporcionada en las plantillas contribuyó a que cada desarrollador fuera organizado y siguiera las reglas establecidas para formar el proceso en su totalidad.

Dificultades y recomendaciones.

Finalmente, el ejercicio obtuvo la retroalimentación de las tareas realizadas. Por parte de los colaboradores no presenciaron dificultades mayores ya que estas no fueron más que dudas que fácilmente se despejaron con la ayuda del jefe de zona.

Como parte de las recomendaciones se sugiere que los componentes que tengan una relación de dependencia se documenten de manera mucho más precisa para facilitar mejor la comprensión, codificación y su posterior funcionamiento a través del estándar de documentación. Además, la aplicación de la practica en contextos más grandes de desarrollo contribuirá a complementar las plantillas de acuerdo al caso particular.

El proyecto concluido fue trabajado en lenguaje java, usando como IDE Netbeans y se encuentra en un repositorio público en Github en el enlace https://github.com/Marcela1396/UdenarColaborativeTeam

CONCLUSIONES

Con la finalización de este proyecto se obtiene la definición de una práctica de desarrollo de software para equipos sin restricciones de tiempo y espacio utilizando los principios de SEMAT.

Mediante el análisis y comprensión del estado del arte y el acercamiento a algunos equipos de desarrollo de software de la ciudad de Pasto, se obtuvo diferentes perspectivas del trabajo distribuido, resaltando sus aspectos positivos y negativos que a su vez fueron el punto de partida de esta investigación para catalogar esta modalidad como una tendencia que se está acoplando a las nuevas necesidades de las organizaciones y que propicia grandes aprendizajes entre sus colaboradores.

Al definir la práctica de desarrollo de software que abarque estas circunstancias se estableció cuan relevante es coordinar el trabajo en equipo cuando se hace de manera distribuida, utilizar como apoyo las diversas herramientas tecnológicas existentes y generar que cada participante ponga a prueba sus habilidades o destrezas para cumplir con cada objetivo que se le plantee.

Por otra parte, la lectura de información bibliográfica contribuyó a mantener una buena documentación respecto al tema, indagar mejor la situación de los equipos de trabajo sin restricciones de tiempo y espacio y caracterizar este tipo de contextos, como una actividad cada vez más común y acorde a las nuevas necesidades que la sociedad impone día con día.

El estándar Essence propuesto por SEMAT permitió que la práctica se consolidará en base a su núcleo de elementos tal como son los alfas de las cuales se abarcaron dos (Requisitos y Sistema Software), los espacios de actividad y competencias permitiendo organizar las actividades que la investigación trae consigo para posteriormente representarla mediante su sintaxis gráfica.

Finalmente se asignó un nombre a la práctica de acuerdo con las alfas que relaciona y la evolución de los estados que estos presentan, tal y como propone la tesis Doctoral del Ingeniero Alexander Barón Salazar, codirector del proyecto, concluyendo que el nombre pertinente sería el de “Implementación Global del Sistema Software”.
Gracias a la aplicación de la práctica en un contexto real, se logró determinar qué tan claras fueron las plantillas para un equipo de desarrolladores y como las actividades propuestas fueron llevadas a cabo exitosamente para lograr el objetivo final del proyecto. Frente a ello, los resultados obtenidos fueron positivos resaltando un alto nivel de aceptación por parte del grupo de participantes. Se espera detallar mas concretamente la plantilla de Funcionalidades agregándole

una casilla de prioridad a cada elemento de la lista y a la plantilla de Procesos la especificación de las fechas límites de entrega para tener una idea de la importancia que tiene efectuar su realización en un determinado periodo de tiempo.

Como parte del trabajo futuro se resalta la posibilidad de adoptar la totalidad de la práctica en equipos de desarrollo de mayor tamaño, donde sus condiciones geográficas y horarias sean más diversificadas, contando así con una visión mucho más amplia sobre aspectos por optimizar.

Adicionalmente como la práctica abarcó únicamente la etapa de codificación, quedaron pendientes muchos aspectos de la parte de gestión en la conformación de equipos que podrían complementarse con un estudio posterior y propiciar una relación fuerte entre ambas que contribuyan en su totalidad a la mejora continua del trabajo remoto y la interacción eficaz de los integrantes.

RECOMENDACIONES

La investigación trajo consigo muchos aspectos positivos durante todo su proceso. Se expusieron fases del proyecto a través de presentaciones públicas y publicaciones en congresos nacionales e internacionales, lo cual permitieron explorar de manera más amplia áreas del conocimiento, junto con la apropiación de nuevas temáticas que produjeron grandes alcances dentro de la experiencia académica y los cuales sirvieron como retroalimentación y mejora continua de cada producto obtenido.

Se pretende que la práctica se valide en contextos mucho más amplios, con un mayor número de participantes y con diferencia geográfica y horaria permitiendo así que equipos de desarrollo de software global la utilicen en su forma de trabajo y puedan medir sus resultados de manera efectiva.

Siendo la investigación enfocada a mejorar la organización y coordinación en equipos de desarrollo distribuidos se busca fomentar el trabajo colaborativo como una iniciativa de gran relevancia no solo en la industria sino en la academia, resaltando que de esta manera se pueden fortalecer lazos de comunicación, integración y cooperación entre los participantes y donde las ganas de aprender se vean reflejadas en una ayuda mutua y colectiva.

Se espera además integrar la práctica de desarrollo de software para equipos distribuidos con otra relacionada a la gestión del proyecto de tal manera que conjuntamente se complementen y ambas se consoliden en base al núcleo de SEMAT.

Como trabajo futuro, sería muy relevante construir una plataforma que enlace lo propuesto por SEMAT con las actividades que involucra la práctica, facilitando un seguimiento permanente de los factores de desarrollo tales como el tiempo invertido, la evolución de trabajo y el grado de satisfacción que existe dentro de un equipo de desarrollo de software.

Finalmente se pretende adquirir mayor experiencia en el campo de la investigación y en el área de Ingeniería de Software con el fin de contribuir a plantear prácticas que sirvan para solventar los problemas existentes y sean aplicables a las nuevas tendencias y retos propuestas por la comunidad de SEMAT.

BIBLIOGRAFÍA

[1] R. S. Pressman, Ingenieria de Software, Un enfoque práctico, México: Mc Graw Hill, 2010.

[2] «Introducción a la Ingeniería del Software Metodologías de Desarrollo de Software,» [En línea]. Available: https://uvirtual.unet.edu.ve/file.php/419/Metodologias_de_desarrollo_de_Software_IS.pdf.

[3] G. FariñoR., «Modelo Espiral de un proyecto de desarrollo de software,» 2011. [En línea]. Available: http://www.ojovisual.net/galofarino/modeloespiral.pdf.

[4] La Prole, «Modelo de Desarrollo Concurrente,» 6 Diciembre 2012. [En línea]. Available: http://laprole431.blogspot.com.co/.

[5] Gestion de Proyectos Software, Google Sites, «Gestion de Proyectos Software,» [En línea]. Available: https://sites.google.com/site/gestiondeproyectossoftware/unidad-2-calidad-de-software/2-2-1-psp-y-tsp.

[6] HungriaBerbesi, «Modelos prescriptivos, Proceso Unificado y Desarrollo Ágil.,» [En línea]. Available: https://implantacionsist.files.wordpress.com/2010/10/modelos-prescriptivos-y-desarrollo-agil.pdf.

[7] EST. JOSE M. BAUTISTA, «PROGRAMACION EXTREMA XP,» [En línea]. Available: http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.

[8] D. Gonzalez y J. L. Perea. [En línea]. Available: http://javeriana.edu.co/biblos/tesis/ingenieria/Tesis192.pdf.

[9] Oiver Andrés Pérez, «Cuatro enfoques metodológicos para el desarrollo de software : RUP – MSF – XP - SCRUM,» 10 Junio 2011. [En línea]. Available: http://biblioteca.uniminuto.edu/ojs/index.php/Inventum/article/view/9.

[10] L. Fuentes, J. M. Troya y A. Vallecillo, «Desarrollo de Software Basado en Componentes,» [En línea].

[11] Carlos Mario Zapata Jaramillo, «La Esencia de la Ingeniería de Software:,» 12 Diciembre 2012. [En línea]. Available: http://sistemas.unla.edu.ar/sistemas/redisla/ReLAIS/relais-v1-n3-p-71-78.pdf.

[12] SEMAT, «SEMAT,» [En línea]. Available: http://semat.org/what-is-it-and-why-should-you-care-.

[13] Object Management Group, «Essence - Kernel and Language for Software Engineering Methods.,» 2 Diciembre 2015. [En línea]. Available: http://www.omg.org/spec/Essence/1.1/PDF/.

[14] L. D. J. Pinzón, «Representación en el Nucleo de SEMAT de practicas de metodos de desarrollo basados en planes,» 2016. [En línea]. Available: http://www.bdigital.unal.edu.co/52534/1/1152184848.pdf.

[15] F. J. R. MORALES, «PRÁCTICAS, METODOLOGÍAS Y HERRAMIENTAS PARA LA GESTIÓN Y EJECUCIÓN DE PROYECTOS DE DESARROLLO DE SOFTWARE QUE UTILICEN EQUIPOS DE TRABAJO GEOGRÁFICAMENTE DISTRIBUIDOS,» 2008. [En línea]. Available: https://repositorio.uc.cl/bitstream/handle/11534/1420/502313.pdf?sequence=1&isAllowed=y.

[16] J. Fried y D. Heinemeier Hansson, Remote, Office not Required, Empresa Activa, 2013.

[17] Johanna Alvarez, Victor Bravo - Fundación CENDITEL, «METODOLOGÍA PARA EL DESARROLLO COLABORATIVO DE SOFTWARE LIBRE,» 2015. [En línea]. Available: http://www.cenditel.gob.ve//files/u206/MetodologiaSoftwareLibre.pdf.

[18] Ing. Fernando Javier Lage, «AMBIENTE DISTRIBUIDO APLICADO A LA FORMACIÓN/CAPACITACIÓN DE RR HH: UN MODELO DE APRENDIZAJE COOPERATIVO-COLABORATIVO,» 2001. [En línea]. Available: http://sedici.unlp.edu.ar/handle/10915/4058.

[19] S. A. P. Montoya, «El teletrabajo en la industria del desarrollo de software en Antioquia,» 2014. [En línea]. Available: https://repository.upb.edu.co/bitstream/handle/20.500.11912/2510/Reporte.pdf?sequence=1&isAllowed=y.

[20] MINTIC, «Teletrabajo,» [En línea]. Available: http://www.mintic.gov.co/portal/vivedigital/612/w3-propertyvalue-571.html.

[21] Manrique-Losada y Gómez-Álvarez, «Representation of TSP framework into the SEMAT kernel based on the best practices of project management from CMMI-DEV,» pp. 33-39, 2014.

[22] J. O. M. Rengifo, «Equipos de Desarrollo de software, sus practicas representadas en SEMAT,» 2015. [En línea]. Available: http://www.bdigital.unal.edu.co/51884/1/71269158.2015.pdf.

[23] MinTIC, MinTrabajo, Vive Digital, «LIBRO BLANCO, EL ABC DEL TELETRABAJO EN COLOMBIA,» 2012. [En línea]. Available: http://www.teletrabajo.gov.co/622/articles-8228_archivo_pdf_libro_blanco.pdf.

[24] L. H. R. SATIZÁBAL, «EL TELETRABAJO: LOS BENEFICIOS DE UNA FORMA DE ORGANIZACIÓN LABORAL MODERNA,» 2014. [En línea]. Available: http://repository.unimilitar.edu.co/bitstream/10654/11993/1/EL%20TELETRABAJO.pdf.

[25] MinTIC, «ESTUDIO PENETRACIÓN TELETRABAJO 2016,» Noviembre 2016. [En línea]. Available: http://www.teletrabajo.gov.co/622/articles-16887_archivo_pdf_estudio_teletrabajo_2016.pdf.

[26] NACIONES UNIDAS - UNCTAD, «INFORME SOBRE LA ECONOMÍA DE LA INFORMACION 2012 - La industria del software y los países en desarrollo,» 2012. [En línea]. Available: http://unctad.org/es/PublicationsLibrary/ier2012_es.pdf.

[27] *Diferencia entre Teletrabajo y Trabajo 3.0.* [Película]. 2016.

[28] MinTIC, MinTrabajo, Vive Digital, «PRIMER ESTUDIO DE TRABAJO 3.0 EN COLOMBIA,» 2015. [En línea]. Available: http://www.teletrabajo.gov.co/622/articles-13905_archivo_pdf_resultados_2015.pdf.

[29] S. A. Sarmiento, «¿Qué son el Crowdsourcing y el Crowdfunding?,» 21 Febrero 2017. [En línea]. Available: http://www.youngmarketing.co/que-es-y-como-ha-transformado-el-crowdsourcing-al-emprendimiento/.

[30] J. Quaglia, «¿Qué es el Crowdsourcing?,» 20 Agosto 2010. [En línea]. Available: https://www.gestiopolis.com/que-es-crowdsourcing/.

[31] D. Rodríguez, R. Priano, N. G. Charczuk, R. Martínez y R. García, «Medidas de interacción en espacios virtuales para trabajo colaborativo,» 2015. [En

línea]. Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/46341/Documento_completo.pdf?sequence=1.

[32] R. Herranz, N. Mamoghli, S. Yazyi, J. M. Vera, S. Arauzo, V. Ratón y M. Salas, «Scrum Distribuido, Trabajo de Investigación,» 14 Junio 2011. [En línea]. Available: http://www.scrummanager.net/files/scrum_distribuido.pdf.

[33] Software Guru, «Desarrollo Global de Software,» [En línea]. Available: https://sg.com.mx/content/view/1038. [Último acceso: 10 Febrero 2018].

[34] *Desarrollo Global de Software.* [Película]. T3chFest, 2014.

[35] S. Misra, R. Colomo-Palacios, T. Pusatli y P. Soto-Acosta, «A discussion on the role of people in global software development,» *Tehnički vjesnik,* pp. 525-531, 2013.

[36] M. Resendiz, «Kernel Case,» [En línea]. Available: http://kernelcase.mx/diferencia-entre-equipos-distribuidos-y-dispersos/.

[37] E. Cesar, «¿Qué es GSD - Desarrollo de Software Global?,» 6 Abril 2015. [En línea]. Available: http://www.cxo2cio.com/2015/04/que-es-gsd.html.

[38] R. Pressman y B. Maxim, « Human Aspects of Software Engineering,» de *Software Engineering: A Practitioner's Approach*, 2015, pp. 87-102.

[39] P. Roger y M. Bruce, «Human Aspects of Software Engineering,» de *Software. Engineering: A Practitioner's Approach.* , 2015.

[40] S. Y. Chadli, A. Idri, J. L. F. Aleman y J. N. Ros, «Identifying Risks of Software Project Management in Global Software Development: An Integrative Framework,» de *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, Agadir, Morocco, 2016.

[41] J. Verner, O. Brereton, B. Kitchenham, M. Turner y M. Niazi, «Risks and risk mitigation in global software development: A tertiary study,» *Information and Software Technology,* pp. 54-78, 2014.

[42] A. A. Khan y J. Keung, «Systematic review of success factors and barriers for software process improvement in global software development,» *IET Software,* pp. 1-11, 2016.

[43] S. Esquivel, «Scrum en Entornos Geográficamente Dispersos.,» 2016.

[44] Wikipedia, «Yahoo,» Wikipedia, [En línea]. Available: https://es.wikipedia.org/wiki/Yahoo!.

[45] J. F. Brian Drummond, «Yahoo! Distributed Agile: Notes from the World Over,» de *Agile, 2008. AGILE '08. Conferencia*, Toronto, 2008.

[46] J. Sutherland, G. Schoonheim, E. Rustenburg y M. Rijk, «Fully Distributed Scrum: The Secret Sauce for Hyperproductive Offshored Development Teams,» de *Agile 2008 Conference*, 2008.

[47] M. Cristal, D. Wildt y R. Prikladnicki, «Usage of SCRUM Practices within a Global Company,» *IEEE International Conference on Global Software Engineering,* pp. 222-226, 2008.

[48] H. Smits y G. Pshigoda, «Implementing Scrum in a Distributed Software Development Organization,» *Agile 2007,* 2007.

[49] Blender, «Blender,» [En línea]. Available: https://www.blender.org/.

[50] Geeks Company, *Presentación de Slack en español,* Youtube, 2015.

[51] N. Fernández, «Tutorial Slack: qué es, para qué sirve y cómo funciona,» Comunicacion Online, 24 Febrero 2016. [En línea]. Available: http://www.naiarafernandez.com/tutorial-slack-que-es-para-que-sirve-y-como-funciona/.

[52] P. Santamaria, «Slack, aprende a sacarle partido a la herramienta de comunicación de moda,» mobbot, 17 Abril 2015. [En línea]. Available: https://www.nobbot.com/personas/slack-aprende-sacarle-partido-la-herramienta-de-comunicacion-de-moda/.

[53] P. Moya, «Stride, la nueva aplicación de comunicación para equipos que compite con Slack,» Omicrono, 8 Septiembre 2017. [En línea]. Available: https://omicrono.elespanol.com/2017/09/stride-comunicacion-para-equipos-slack/.

[54] M. Pinola, «Cómo utilizar Trello para organizar tu vida casi al completo,» Gizmodo, 2 Agosto 2015. [En línea]. Available: https://es.gizmodo.com/como-organizar-toda-tu-vida-utilizando-trello-1684529913.

[55] Atlassian, «Jira Software,» atlassian, [En línea]. Available: https://es.atlassian.com/software/jira.

[56] Basecamp, «Así es como funciona Basecamp,» Basecamp, [En línea]. Available: https://basecamp.com/how-it-works.

[57] LuchoCastillo, «Conociendo Github,» Github, 17 Octubre 2012. [En línea]. Available: https://github.com/LuchoCastillo/Conociendo-GitHub/blob/master/tutorial/data/introduccion.rst.

[58] J. Sanchez, «Qué es Github y cómo te puede ayudar,» Freelancer, 11 Octubre 2016. [En línea]. Available: https://www.freelancer.com.co/community/articles/github-como-puede-ayudar.

[59] N. Gonzales, «Github vs Bitbucket. Mi opinión personal sobre cual usar.,» Disqus, [En línea]. Available: http://www.nazariglez.com/2013/02/15/github-vs-bitbucket-mi-opinion-personal-sobre-cual-usar/.

[60] Github, «Buenas Practicas de codificacion,» 27 Marzo 2015. [En línea]. Available: https://github.com/kosme10/standards/wiki/Buenas-Practicas-de-codificacion#variables.

[61] J. S. Vanegas, «Buenas practicas en la codificacion,» Juan Sebastian Vanegas, [En línea]. Available: http://undiaparahablar.blogspot.com.co/2010/03/buenas-practicas-en-la-codificacion.html.

[62] HaskellWiki, «Programming guidelines,» 22 Septiembre 2017. [En línea]. Available: https://wiki.haskell.org/Programming_guidelines#File_Format.

[63] JEE, «Buenas prácticas codificación,» JEE, [En línea]. Available: https://infoxicated.wordpress.com/buenas-practicas-codificacion/.

[64] Java Foundations, «Java - Estándares de programación,» 2 Julio 2010. [En línea]. Available: http://javafoundations.blogspot.com.co/2010/07/java-estandares-de-programacion.html.

[65] Metro de Santiago, «Estándares Informáticos para Proyectos de Software,» [En línea]. Available: https://www.metro.cl/licitaciones/descarga/acf922154627f6788918f03c42b12

3cd..

[66] Universidad de los Andes, «Cupidos Uniandes,» Uniandes, [En línea]. Available: https://cupi2.virtual.uniandes.edu.co/ejercicios-del-semestre-apo1/ejercicio-n2.

ANEXOS

ANEXO A. ENCUESTAS EQUIPOS DE DESARROLLO DE SOFTWARE

Población: 62 personas pertenecientes a equipos de desarrollo de software en la ciudad de San Juan de Pasto.

Objetivo General: Determinar la importancia que tiene para los miembros del equipo de desarrollo de software, el trabajar conjuntamente sin ninguna restricción de tiempo y espacio.

Objetivos Específicos:

- Establecer el nivel profesional que tiene los desarrolladores, determinando así su experiencia en la construcción de software.
- Determinar cómo es la interacción del desarrollador con el resto del equipo y las dificultades que se han presentado.
- Establecer si el desarrollador ha trabajado de manera remota.
- Indagar las ventajas y desventajas que el desarrollador piensa sobre trabajar en diferentes contextos de tiempo y espacio.

ANALISIS DE LOS RESULTADOS

La presente investigación trajo consigo en su primera etapa la realización de encuestas con algunos de los equipos de desarrollo de software de la ciudad de Pasto. El instrumento se aplicó cabo durante el periodo septiembre – octubre del año 2017, estableciendo así algunos elementos claves para el estudio que se está efectuando. El total de encuestados fue de 62 personas y las empresas que colaboraron para dicho objetivo fueron:

Tabla A 1. Empresas, centros de desarrollo encuestadas

Equipos de desarrollo	Recuento	Porcentaje
Centro de Informatica Udenar	16	25,8%
CJTYT	13	21,0%
Institución Universitaria Cesmag	8	12,9%
GRIAS	6	9,7%
Apptividad	4	6,5%
Hashito Apss	4	6,5%
Parquesoft	4	6,5%
JHWeb Pasto	2	3,2%
Otras	5	8,1%

La encuesta se dividió en tres secciones, la primera enfocada en el estado actual del encuestado dentro de la empresa. La siguiente enfocada en la interacción con el equipo de desarrollo de software y finalmente la última dedicada a las diferentes modalidades de trabajo.

ESTADO ACTUAL

1) ¿Cuál es el grado profesional que usted posee?

Según el número de encuestados, la mayor parte de la población se encuentra en su nivel de estudios de pregrado teniendo un porcentaje de alrededor del 63%, seguido de un 27% en el grado profesional de especialización. De la población encuestada no se encontró nadie con estudios superiores en Doctorado.

Tabla A 2. Grado Profesional Encuestados

		Recuento	%
Grado profesional	Técnico	4	6,5%
	Pregrado	39	62,9%
	Especialización	17	27,4%
	Maestría	2	3,2%
	Doctorado	0	0,0%

Figura A 1. Grado Profesional Encuestados

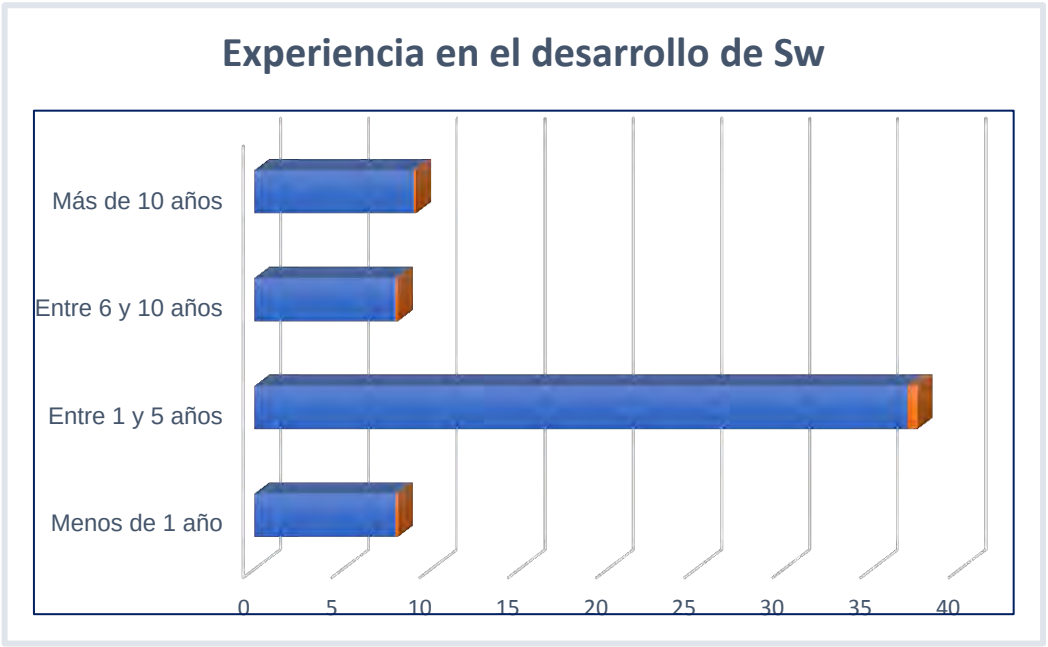
2) ¿Cuánto es el tiempo de su experiencia profesional dentro del desarrollo de software?

De los encuestados, el tiempo de su experiencia profesional desarrollando software oscila entre 1 y 5 años siendo casi un 60% del total de la población. De esto se podría inferir que durante este tiempo el desarrollador puede tener ya un conocimiento más detallado de las herramientas software y de plataformas empleadas para dicho propósito.

Tabla A 3. Tiempo Experiencia Profesional Encuestados

		Recuento	%
Experiencia	Menos de 1 año	8	12,9%
	Entre 1 y 5 años	37	59,7%
	Entre 6 y 10 años	8	12,9%
	Más de 10 años	9	14,5%

Figura A 2.Tiempo Experiencia Profesional Encuestados

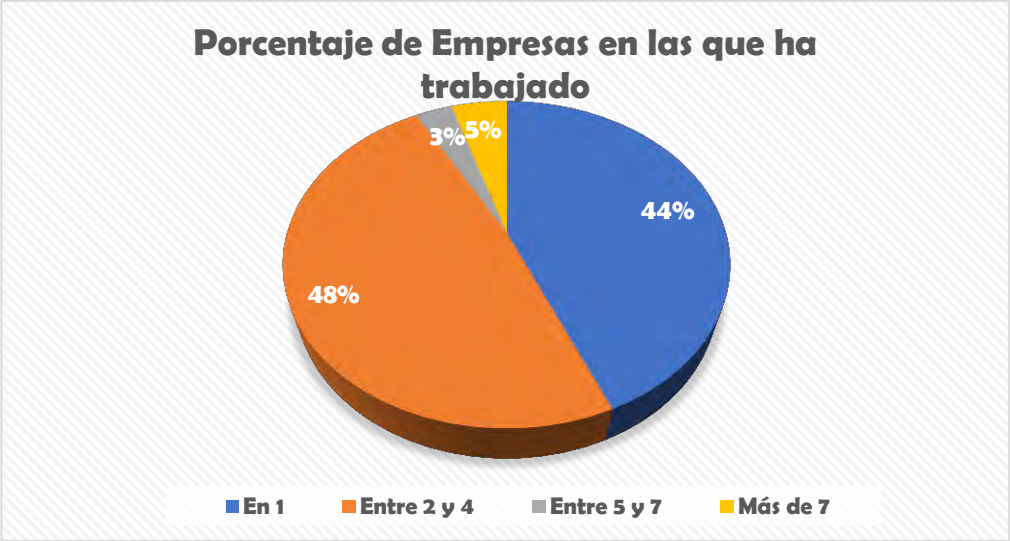


3) ¿En cuántas empresas o equipos de desarrollo de software ha trabajado usted?

Tabla A 4. Número de Empresas Laboradas Encuestados

		Recuento	%
Número de Empresas en las que haya laborado	En 1	27	43,5%
	Entre 2 y 4	30	48,4%
	Entre 5 y 7	2	3,2%
	Más de 7	3	4,8%

Figura A 3. Número de Empresas Laboradas Encuestados



Según los datos recolectados, el número de empresas de desarrollo de software en los cuales la población ha trabajo oscila principalmente entre 2 y 4 con un 48,4%, seguido de un 43,5% en los cuales solo han laborado en una empresa.

Estos porcentajes indican que la mayor parte de los encuestados ya han tenido experiencia laboral en más de una empresa, de lo que se podría inferir que pueden tener más conocimiento sobre formas y ambientes de trabajo en los que hayan interactuado.

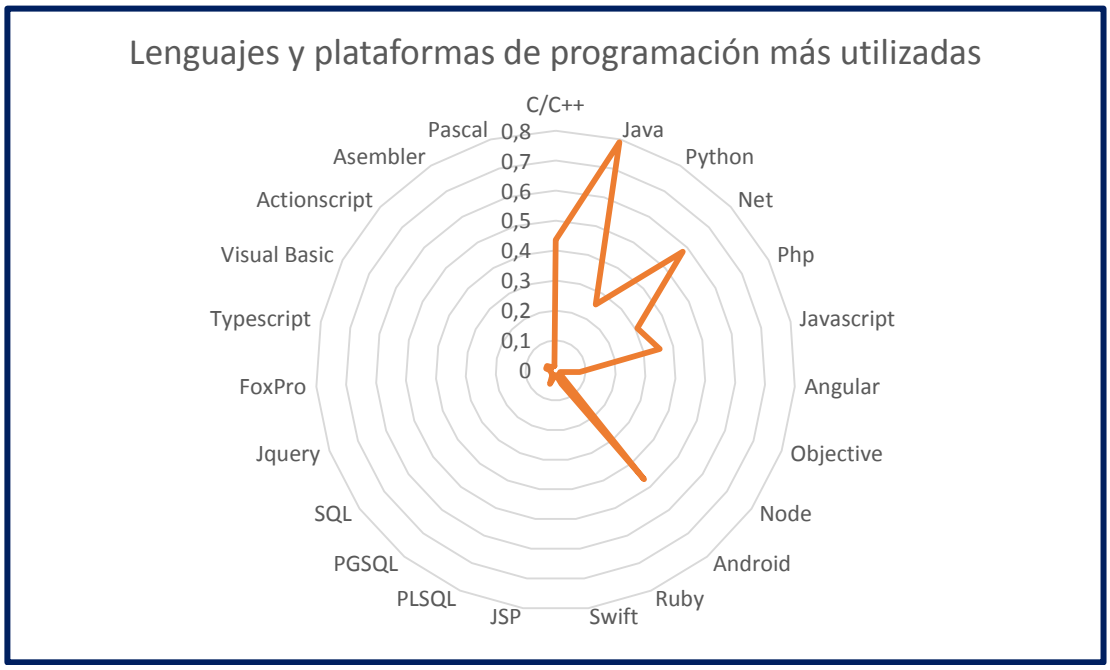
4) ¿Qué lenguaje(s) de programación o plataformas usted maneja?

A los encuestados se les mostro una posible lista de lenguajes y plataformas de programación, de los cuales se obtuvieron los siguientes resultados.

Tabla A 5. Lenguajes y plataformas de programación que manejan encuestados

Lenguajes de Programación	Recuento	%
C/C++	27	43,5%
Java	49	79,0%
Python	16	25,8%
.Net	36	58,1%
Php	19	30,6%
Javascript	22	35,5%
Angular	5	8,1%
Objective	1	1,6%
Node	1	1,6%
Android	29	46,8%
Ruby	3	4,8%
Swift	1	1,6%
JSP	1	1,6%
PLSQL	3	4,8%
PGSQL	1	1,6%
SQL	1	1,6%
Jquery	1	1,6%
FoxPro	1	1,6%
Typescript	2	3,2%
Visual Basic	2	3,2%
Actionscript	1	1,6%
Asembler	1	1,6%
Pascal	1	1,6%

Figura A 4. Lenguajes y plataformas de programación que maneja encuestados.



Según las preferencias de los encuestados, Java es el lenguaje de programación más utilizado por las empresas de desarrollo de software alcanzando casi un 80% de uso. Como framework más empleado esta .Net llegando a un 60% aproximadamente y como plataforma de desarrollo para aplicaciones móviles encontramos a Android con un 47% de aceptación.

INTERACCIÓN CON EL EQUIPO DE DESARROLLO DE SOFTWARE

A continuación, se presentan los resultados de las preguntas relacionadas acerca de la interacción del encuestado con su equipo de desarrollo de software.

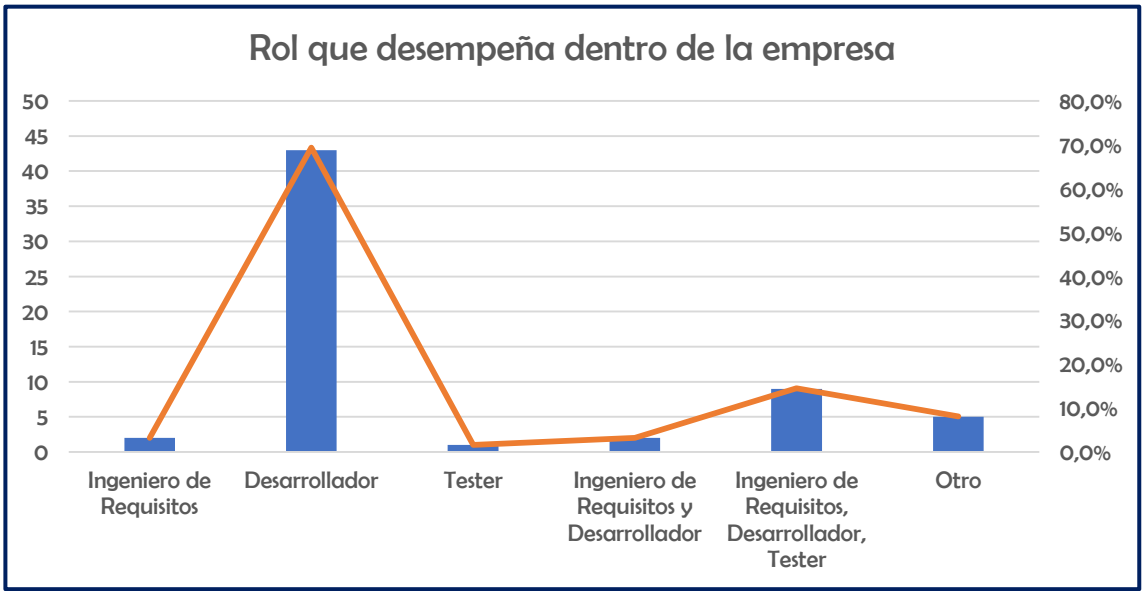
5. ¿Qué rol desempeña dentro del equipo de desarrollo de software?

Tabla A 6. Rol que desempeña el encuestado dentro del equipo

		Recuento	%
Rol que desempeña	Ingeniero de Requisitos	2	3,2%
	Desarrollador	43	69,4%
	Tester	1	1,6%
	Ingeniero de Requisitos y Desarrollador	2	3,2%
	Ingeniero de Requisitos, Desarrollador, Tester	9	14,5%
	Otro	5	8,1%

Según la información recolectada, el rol más común de los encuestados fue el de desarrollador con alrededor de un 70%. Sin embargo, cabe resaltar que un porcentaje del 14.5% de la población afirmo que desenvuelve los tres roles dentro del equipo de desarrollo (Ingeniero de Requisitos, Programador y Tester), lo que significa que sus funciones varían de acuerdo a las necesidades del proyecto.

Figura A 5. Rol que desempeña el encuestado dentro del equipo



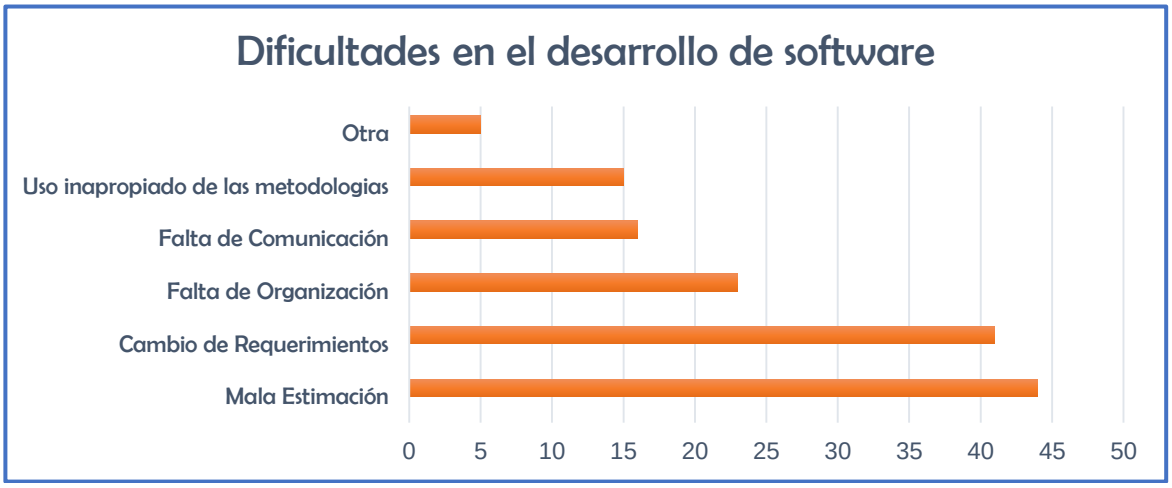
6. ¿Cuáles son las principales dificultades que usted ha enfrentado dentro del equipo de desarrollo y la realización de un proyecto software?

Esta pregunta busca la finalidad de encontrar a través del encuestado cuales cree que son las principales dificultades al momento de desarrollar software dentro de un equipo. Frente a esto se postularon 6 posibles opciones.

Tabla A 7. Dificultades dentro del equipo de desarrollo.

Dificultades	Recuento	%
Mala Estimación	44	71,0%
Cambio de Requerimientos	41	66,1%
Falta de Organización	23	37,1%
Falta de Comunicación	16	25,8%
Uso inapropiado de las metodologías	15	24,2%
Otra	5	8,1%

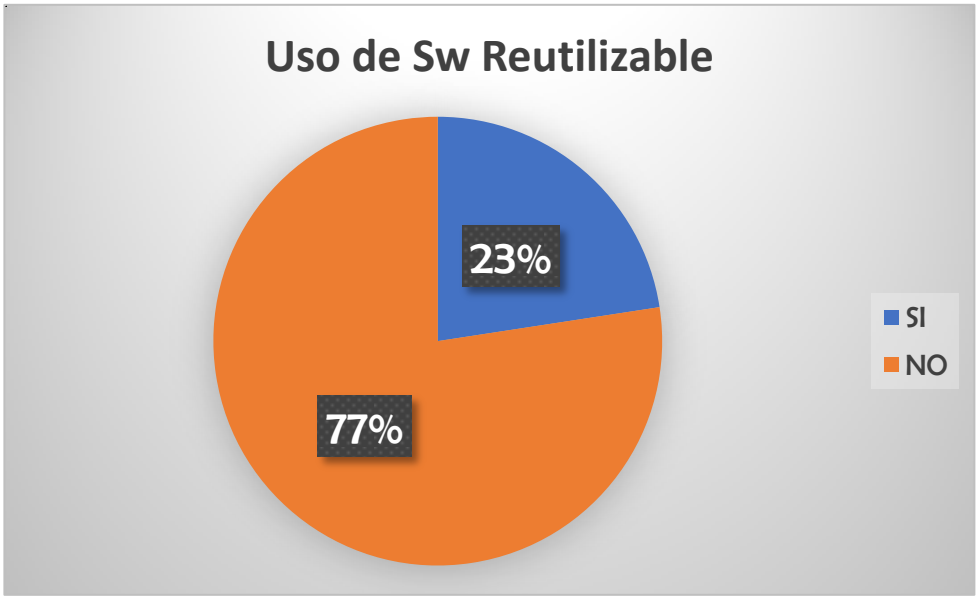
Figura A 6. Dificultades dentro del equipo de desarrollo



Los resultaron arrojaron que la mala estimación de tiempo para el desarrollo de software es el mayor problema presentado con un porcentaje del 71%, seguido por el cambio constante de requerimientos durante el proyecto con un 66,1%. Por otra parte, un 8.1% de la población opino que otras de las dificultades podrían ser la falta de un buen presupuesto, el poco conocimiento para trabajar en equipo y la toma de decisiones de forma apresurada.

7.) ¿Ha desarrollado algún componente software que pueda ser reutilizable por otras personas?

Figura A 7. Uso de Software Reutilizable



De acuerdo a la información recolectada solo el 23% de la población ha desarrollado algún componente software reutilizable tanto para la empresa para la que trabajan o equipo al que pertenecen como también de uso libre para cualquiera que lo necesite.

MODALIDADES DE TRABAJO

8. ¿Ha trabajado usted en desarrollo de software de modo freelance?

Tabla A 8. Trabajo freelance en desarrollo de software

		Recuento	%
Trabajo freelance en el desarrollo de Sw	SI	27	43,5%
	NO	35	56,5%

Figura A 8. Trabajo freelance en desarrollo de software

El Trabajo Freelance es un trabajo independiente que no exige la presencia física del empleado dentro de la empresa y que además puede desarrollar sus actividades desde casa o el lugar que disponga.

Según la información recolectada por parte de los encuestados, el 44% han optado por adoptar esta modalidad. Frente a los que respondieron afirmativamente se optó por preguntarles que efecto había traído para ellos ser freelance y por qué; con esto se obtuvo que de ese porcentaje el 93% dijeron que los resultados fueron netamente positivos resaltando como ventajas:

- Eres tú propio jefe y desarrollas tus propias ideas al ritmo propio.
- Trabajo desde casa, manejando el horario que se desee y distribuyéndolo de la mejor manera posible.
- Comodidad y mayor flexibilidad para laborar.
- Evitar movilizaciones hacia un lugar de trabajo establecido lo que propiciaría un ahorro en dinero y tiempo.

Figura A 9. Resultados de trabajo freelance

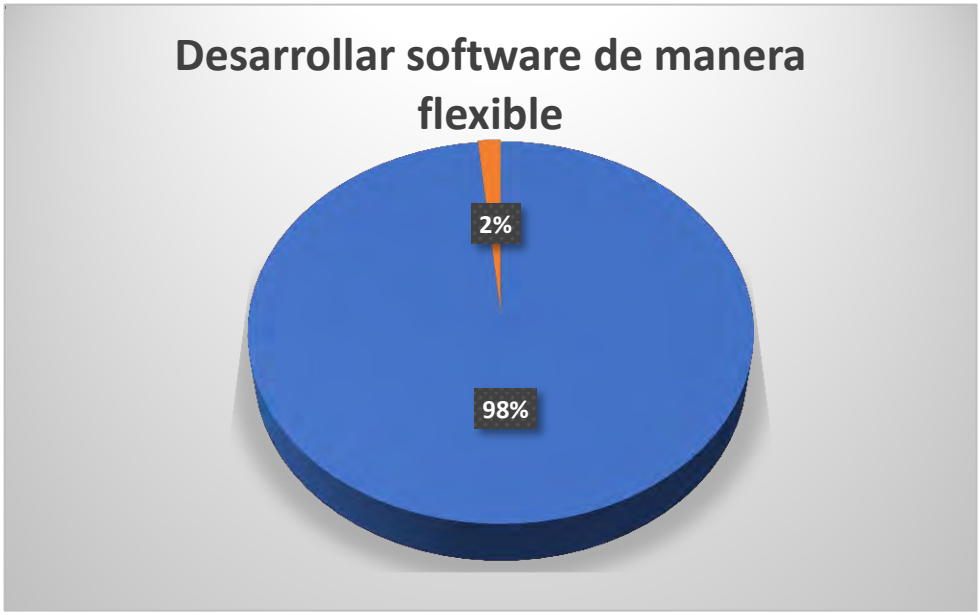
Adicionalmente se discutió sobre la metodología o forma de trabajo que hayan utilizado cuando son freelance, resaltando el uso de SCRUM y XP; por parte de herramientas de administración de proyectos destacaron GitHub y Trello y por el lado de comunicación e interacción a Skype.

9) ¿Le gustaría desarrollar software sin importar el lugar desde donde lo haga o el tiempo en que usted se dedique, teniendo así una flexibilidad de horarios y que a su vez sea remunerado?

Tabla A 9. Desarrollo de Software de manera flexible

		Recuento	%
Desarrollo De Sw de manera flexible	SI	60	96,8%
	NO	2	3,2%

Figura A 10. Trabajo freelance en desarrollo de software



Esta pregunta permitió averiguar si los encuestados les gustaría tener una nueva forma de trabajo mejor establecida y organizada para aquellos casos en que se desarrolla software a distancia sin importar el lugar donde se realice este tipo de actividades o el tiempo en que usted se dedique, teniendo un porcentaje de aceptación del 97%.

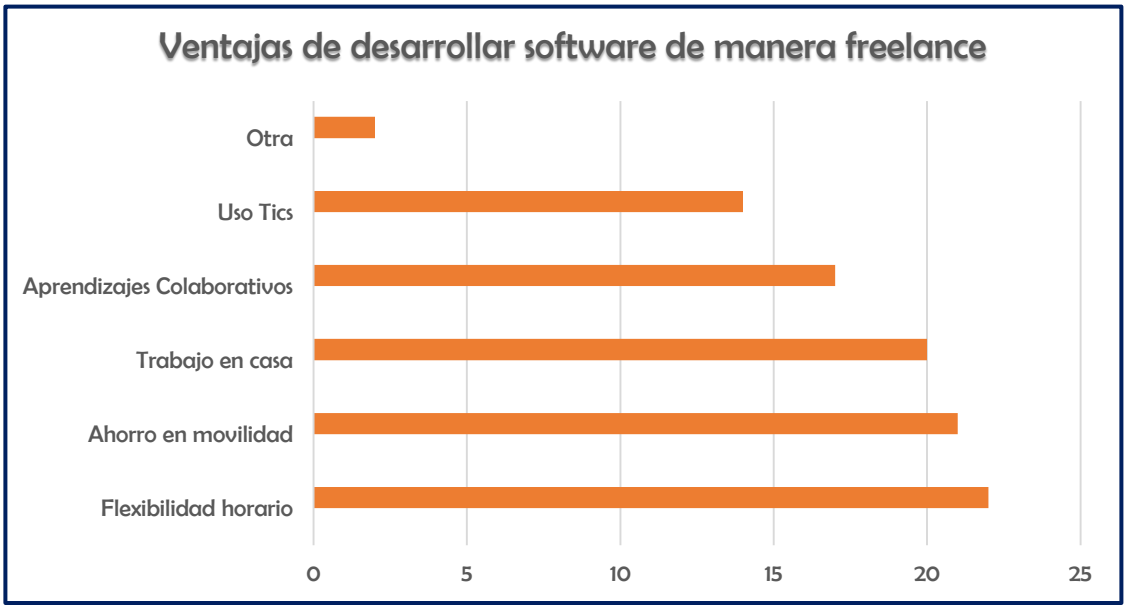
10) ¿Qué ventajas considera usted, tiene el desarrollo de software de modo freelance?

De los encuestados que respondieron que efectivamente habían trabajado de manera freelance (43,5%) se determinó las principales ventajas que ellos consideraban al hacerlo de esta manera, tal y como se visualiza en la siguiente tabla.

Tabla A 10. Ventajas de trabajar de manera freelance.

Ventajas	Recuento	%
Flexibilidad horario	22	81,5%
Ahorro en movilidad	21	77,8%
Trabajo en casa	20	74,1%
Aprendizajes Colaborativos	17	63,0%
Uso Tics	14	51,9%
Otra	2	7,4%

Figura A 11. Ventajas de trabajar de manera freelance.



A continuación, se presentó una lista de opciones a los encuestados para que den su opinión sobre cuales consideran ser las ventajas de desarrollar software de manera freelance. Esta pregunta fue de opción múltiple.

Entre las ventajas que destacaron se encuentran:

- Flexibilidad de horarios con un 81,5%
- Ahorro en movilidad con un 77,8%
- Trabajo desde casa con un 74,1%.

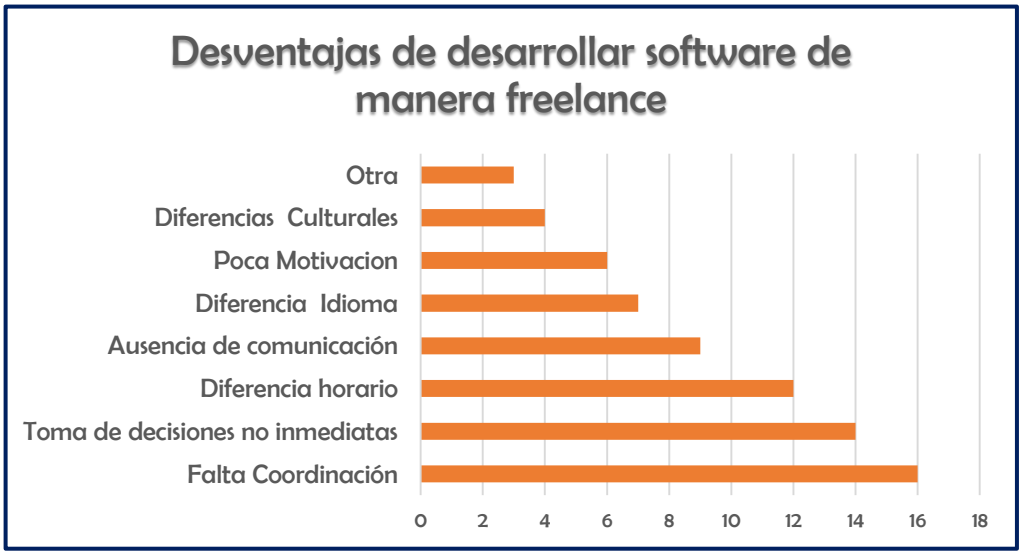
11) ¿Qué desventajas considera usted, tiene el desarrollo de software de modo freelance?

De igual manera se indago por las desventajas que consideran al aplicar esta modalidad.

Tabla A 11.Desventajas de trabajar de manera freelance.

Desventajas	Recuento	%
Falta Coordinación	16	59,3%
Toma de decisiones no inmediatas	14	51,9%
Diferencia horario	12	44,4%
Ausencia de comunicación	9	33,3%
Diferencia Idioma	7	25,9%
Poca Motivación	6	22,2%
Diferencias Culturales	4	14,8%
Otra	3	11,1%

Figura A 12. Desventajas de trabajar de manera freelance.



A continuación, se presentó una lista de opciones a los encuestados para que den su opinión sobre cuales consideran ser las desventajas de desarrollar software de manera freelance. Esta pregunta fue de opción múltiple.

Entre las desventajas que destacaron se encuentran:

- Falta de coordinación con un 59,3%
- Toma de decisiones no inmediatas con un 51,9%
- Diferencia horaria con un 44,4%

ANEXO B. PLANTILLAS USADAS PARA LA APLICACIÓN DE LA PRÁCTICA EN UN CONTEXTO REAL

Tabla B 1. Caso práctico. Plantilla de Funcionalidades.

Funcionalidades	
Nombre del sistema:	Sistema para Máquina Expendedora de productos.
Descripción:	Se desea construir una aplicación que permita simular el uso, por parte de un cliente, de una máquina expendedora de alimentos, la cual contiene 4 productos. La aplicación permite al usuario (o cliente de la máquina) añadir monedas a la máquina para tener un crédito de compra, comprar productos, recibir el cambio cuando haya terminado una compra y conocer la información sobre las compras que ha realizado en la máquina.
Características:	1. Visualizar la información de los productos. 2. Agregar crédito (monto de dinero) a la máquina. 3. Comprar un producto. 4. Calcular la cantidad total de unidades compradas (de todos los productos). 5. Calcular el valor total por las compras realizadas. 6. Calcular el porcentaje de disponibilidad de la máquina. 7. Consultar el producto más comprado (del cual ha comprado un mayor número de unidades). 8. Calcular el valor de la donación total al FOPRE por las compras realizadas. 9. Conocer el valor de la donación al FOPRE por tipo de producto. 10. Conocer las unidades compradas de productos FOPRE y por tipo de producto. 11. Terminar la compra.
Información Adicional:	Proyecto Estudiantil
Número de Procesos:	1
Fecha de Creación:	4 - Abril - 2018
Observaciones:	Ninguna.
Definida por:	Marcela Guerrero Tomado de Caso estudio- Ejercicio de Semestre Máquina Expendedora- Uniandes.

Tabla B 2. Caso práctico. Plantilla Procesos.

Procesos	
Id:	PR-01
Nombre:	Proceso de compra y venta de un producto de la Máquina Expendedora.
Descripción:	Permite simular el uso de una máquina expendedora de alimentos, facilitando su compra de acuerdo a un monto establecido y al dinero ingresado por el usuario.
Número de Componentes:	4
Nivel de Prioridad:	Alta
Jefe de Zona Asignado:	Luis España
Zona Asignada:	PAS-04
Fecha de Creación:	4 – 04 - 2018
Fecha de Ejecución:	07 – 04 – 2018
Fecha estimada de entrega del Proceso integrado:	15 – 04 - 2018
Duración Estimada.	10 días
Observaciones:	Trabajo Grupal Distribuido
Definido por:	Marcela Guerrero

Tabla B 3. Plantilla Componente Máquina Expendedora

<table>
<tr><th colspan="2">Componentes</th></tr>
<tr><td>Id:</td><td>COM-01</td></tr>
<tr><td>Nombre:</td><td>Componente Máquina Expendedora</td></tr>
<tr><td>Descripción:</td><td>La máquina expendedora dispone de cuatro productos de los cuales podrán ser comprados por un usuario de acuerdo al dinero que este ingrese.</td></tr>
<tr><td>Característica</td><td>- La máquina puede contener máximo 10 unidades de cada producto. Cuando se ejecuta la aplicación, la máquina expendedora se inicializa con esta capacidad.

- La máquina permite que se done un porcentaje del valor recaudado, por la compra de algunos de sus productos, al FOPRE (Fondo de Programas Especiales). El aporte al FOPRE corresponde al 6% del valor del producto. No todos los productos de la máquina donan al FOPRE.</td></tr>
<tr><td>Entradas:</td><td>Constructor
Constructor de la Maquina inicializando cuatro productos. No recibe parámetros.
Cada producto recibe como parámetros: identificador, nombre y precio.

Métodos

<table>
<tr><th>Nombre del Método o función</th><th>Parámetros de Entrada</th></tr>
<tr><td>darProducto</td><td>Identificador del producto</td></tr>
<tr><td>agregarMoneda</td><td>Denominación de la moneda.</td></tr>
<tr><td>comprarProducto</td><td>Identificador del producto
<code>pIdentificador != null
&&
pIdentificador != "".</code></td></tr>
<tr><td>darDonacionPorTipo</td><td>Tipo de producto</td></tr>
<tr><td>darCantidadUnidadesDisponibles</td><td>Tipo de producto</td></tr>
</table>
</td></tr>
</table>

Salidas Esperadas:	Métodos

Nombre del Método o función	Parámetros de Retorno
darValorCredito	Valor del crédito disponible en la máquina.
darProducto	Producto con el identificador recibido por parámetro
agregarMoneda	True si pudo agregar la moneda, false si superaba el valor máximo
comprarProducto	True si se realiza la compra, false si no hay crédito suficiente o no había unidades disponibles.
terminarCompra	Monto de dinero correspondiente al cambio.
darCantidadTotalUnidadesCompradas	Cantidad total de unidades compradas de la máquina.
darValorTotalCompras	Valor total de las compras de la máquina.
darPorcentajeDisponibilidad	Porcentaje de disponibilidad de la máquina.
darDonacionPorTipo	Total de donación al FOPRE del tipo de producto.
darCantidadUnidadesDisponibles	La suma de las unidades compradas de los productos FOPRE del tipo dado.
darProductoMasComprado	Producto más

		comprado. Si hay dos productos con la mayor cantidad de unidades compradas, se retorna el primero encontrado.
	darDonacionTotal	Donación total al FOPRE.

Diccionario de datos:

Variables

Nombre de Variable	Tipo de Variables	Descripción
producto1 producto2 producto3 producto4	Producto	Relaciona el producto.
pIdentificador	String	Identificador del producto
pMoneda	Int	Denominación de la moneda
pTipo	char	Tipo de producto. C = Comida B = Bebida

Diccionario de datos:

Nombre de Variable	Descripción
cantidadTotalUnidadesCompradas	La suma de las unidades compradas de todos los productos.
$cantidadTotalUnidadesCompradas = \sum_{i=1}^{4} cantidadUnidadesCompradasProducto_i$	
valorTotalCompras	La sumatoria de la cantidad de unidades compradas de cada producto, multiplicada por el precio del mismo.
$valorTotalCompras = \sum_{i=1}^{4} cantidadUnidadesCompradasProducto_i * precioProducto_i$	
porcentajeDisponibilidad	Representa el porcentaje de espacios disponibles (con respecto a la capacidad total de la máquina), suponiendo que cada unidad de producto ocupa un espacio dentro de la máquina.

Donde:

$$porcentajeDisponibilidad = 100 - (\frac{\sum_{i=1}^{4} cantidadUnidadesDisponiblesProducto_i}{CAPACIDAD * cantidadProductos} * 100)$$

-CAPACIDAD es la cantidad máxima de unidades que puede contener la máquina por cada producto.

-cantidadProductos es el número de productos que maneja la maquina (es decir, 4).

Métodos

Nombre de Método o función	Detalle
darValorCredito	Retorna el valor del crédito disponible en la máquina.
darProducto	Retorna un producto siempre y cuando exista alguno con el identificador dado.
agregarMoneda	Agrega la moneda al crédito.
comprarProducto	Compra un producto siempre y cuando exista alguno con el identificador dado. Si hay crédito y unidades suficientes se realiza la compra y se cambia el valor del crédito actual.
terminarCompra	Calcula el monto de dinero que se debe dar como cambio y reinicia el crédito.
darCantidadTotalUnidadesCompradas	Calcula la cantidad total de unidades compradas de la máquina.
darValorTotalCompras	Calcula el valor total de las compras de la máquina.
darPorcentajeDisponibilidad	Calcula el porcentaje de disponibilidad de la máquina.
darDonacionPorTipo	Retorna el valor total de la donación al FOPRE de productos del tipo especificado.
darCantidadUnidadesCompradasFopre	Retorna el total de unidades FOPRE compradas para un tipo específico.
darProductoMasComprado	Retorna el producto más comprado.
darDonacionTotal	Retorna el valor total de la donación al FOPRE.

Interrelación:	Componente Producto Componente Monto
Conexiones a bases de datos:	Ninguna
Diseño del Componente:	SI (Ver Diagrama de clases)
Designado a:	Daniel Tutistar
Rol que cumple:	Desarrollador
Zona Responsable	PAS – 04
Proceso:	PR-01
Nivel de Prioridad:	Alta
Tiempo Estimado:	8 días
Fecha de Inicio:	07 – 04- 2018
Fecha estimada de entrega:	13 – 04 - 2018
Observación	Ninguna
Contrariedad	Ninguna
Definido por:	Marcela Guerrero.

Tabla B 4. Plantilla Componente Producto

Componentes	
Id:	COM-02
Nombre:	Componente Producto
Descripción:	Representa un producto de la máquina expendedora.

Características:

Los productos que contiene la máquina se presentan en la siguiente tabla:

Nombre	Identificador	Precio	Tipo	FORPRE
Papa Natural Margarita	A1	$1300	Comida	SI
Jugo Hit	A2	$2000	Bebida	SI
Chocolatina Jet	B1	$450	Comida	NO
Galletas Festival	B2	$800	Comida	NO

Entradas:

Constructor

Cada producto recibe como parámetros: identificador, nombre, precio, fopre y tipo.

Parámetros de entrada	Tipo	Descripción
pIdentificador	String	Identificador del producto
pNombre	String	Nombre del producto
pPrecio	Double	Precio del producto pPrecio >= 50
pFopre	Boolean	Indicador si el producto dona parte de sus ganancias al FOPRE.
pTipo	Char	Tipo del producto. C = Comida B = Bebida

- Se debe inicializar los nuevos atributos con la información recibida por parámetros
- Se inicializa la cantidad de unidades compradas en 0.
- Se inicializa la cantidad de unidades disponibles con la capacidad.

Métodos:

Ninguno recibe parámetros de Entrada

Salidas

Métodos

<table>
<tr><td>Esperadas:</td><td>
<table>
<tr><th>Nombre del Método o función</th><th>Variables de retorno</th></tr>
<tr><td>darIdentificador</td><td>Identificador del producto.</td></tr>
<tr><td>darNombre</td><td>Nombre del producto.</td></tr>
<tr><td>darPrecio</td><td>Precio del producto.</td></tr>
<tr><td>darTipo</td><td>Tipo del producto.</td></tr>
<tr><td>darCantidadUnidadesDisponibles</td><td>Cantidad de unidades disponibles del producto.</td></tr>
<tr><td>esFopre</td><td>Retorna true si el producto está asociado al FOPRE, false en caso contrario.</td></tr>
<tr><td>comprar</td><td>True si se realiza la compra, false en caso de no haber disponibilidad.</td></tr>
<tr><td>calcularDonacionFopre</td><td>Donación al FOPRE</td></tr>
</table>
</td></tr>
<tr><td>Interrelaciones:</td><td>Ninguna</td></tr>
</table>

Diccionario de datos

Variables

Nombre de Variable	Tipo	Descripción
identificador	String	Identificador del producto
nombre	String	Nombre del producto
precio	double	Precio del producto
cantidadUnidadesDisponibles	Int	Cantidad de unidades disponibles del producto.
cantidadUnidadesCompradas	Int	Cantidad de unidades compradas del producto.
tipo	Char	Tipo del producto
fopre	Boolean	Indicador que determina si el producto aporta o no al FOPRE.

Diccionario de datos

Constantes

Nombre de la Constante	Tipo	Descripción
PORCENTAJE_FOPRE	Double	Constante que representa el porcentaje de donación al FOPRE.
CAPACIDAD	Int	Representa la cantidad máxima de unidades que se puede tener de un producto.

<table>
<tr><td></td><td>Métodos

<table>
<tr><th>Nombre de Método o función</th><th>Detalle</th></tr>
<tr><td>darIdentificador</td><td>Retorna el identificador del producto.</td></tr>
<tr><td>darNombre</td><td>Retorna el nombre del producto.</td></tr>
<tr><td>darPrecio</td><td>Retorna el precio del producto.</td></tr>
<tr><td>darTipo</td><td>Retorna el tipo del producto.</td></tr>
<tr><td>darCantidadUnidadesDisponibles</td><td>Retorna la cantidad de unidades disponibles del producto</td></tr>
<tr><td>esFopre</td><td>Indica si el producto dona parte de sus ganancias al programa FOPRE.</td></tr>
<tr><td>comprar</td><td>-Compra una unidad del producto si hay unidades disponibles.
-Si había unidades disponibles, se disminuye la cantidad de unidades disponibles y aumenta la cantidad de unidades compradas.</td></tr>
<tr><td>calcularDonacionFopre</td><td>Retorna el valor de la donación de este producto al FOPRE.</td></tr>
</table></td></tr>
<tr><td>Conexiones a bases de datos:</td><td>Ninguna</td></tr>
<tr><td>Diseño del Componente:</td><td>SI (Ver Diagramas de clases)</td></tr>
<tr><td>Designado a:</td><td>Camilo Valencia</td></tr>
<tr><td>Rol que cumple:</td><td>Desarrollador</td></tr>
<tr><td>Zona Responsable:</td><td>PAS-04</td></tr>
<tr><td>Proceso:</td><td>PR-01</td></tr>
<tr><td>Nivel de</td><td>Alta</td></tr>
</table>

Prioridad:	
Tiempo Estimado:	8 días
Fecha de Inicio:	07 – 04 -2018
Fecha de estimada de entrega:	13 – 04 - 2018
Observaciones:	Ninguna
Contrariedades:	Ninguna
Definido por:	Marcela Guerrero

Tabla B 5. Plantilla Componente Máquina Monto

<table>
<tr><th colspan="2">Componentes</th></tr>
<tr><td>Id:</td><td>COM-03</td></tr>
<tr><td>Nombre:</td><td>Componente Monto</td></tr>
<tr><td>Descripción:</td><td>Representa un monto.</td></tr>
<tr><td>Características:</td><td>Un monto de dinero está conformado por monedas de las siguientes denominaciones: $50 COP, $100 COP, $200 COP, $500 COP y $1.000 COP. El cambio que le da la máquina al usuario (el monto de dinero que recibe el usuario después de terminar su compra) se debe retornar siempre utilizando la menor cantidad de monedas posible.
- Añadir a la máquina un monto de dinero, llamado crédito, para realizar sus compras. El monto máximo que un usuario puede ingresar a la máquina es de $10.000 COP.
- Al terminar la compra recibe un monto de cambio (dinero que le devuelve la máquina), de acuerdo al dinero ingresado y al valor total de la compra.</td></tr>
<tr><td rowspan="3">Entradas:</td><td>Constructor</td></tr>
<tr><td>Crea un nuevo monto sin monedas.
Se inicializaron todas las cantidades en 0.</td></tr>
<tr><td>Métodos

<table>
<tr><th>Nombre del Método o función</th><th>Parámetros de Entrada</th></tr>
<tr><td>agregarMoneda</td><td>Denominación de la moneda
pMoneda
{MONEDA_50,
MONEDA_100,
MONEDA_200,
MONEDA_500,
MONEDA_1000}.</td></tr>
<tr><td>cambiarValor</td><td>Valor a completar en monedas. pValor >= 0 && pValor%50 == 0.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Salidas Esperadas:</td><td>Métodos

<table>
<tr><th>Nombre del Método o función</th><th>Variables de retorno</th></tr>
<tr><td>darCantidadMonedas50</td><td>La cantidad de monedas de 50.</td></tr>
<tr><td>darCantidadMonedas100</td><td>La cantidad de monedas de 100.</td></tr>
<tr><td>darCantidadMonedas200</td><td>La cantidad de monedas de 200.</td></tr>
<tr><td>darCantidadMonedas500</td><td>La cantidad de monedas de 500.</td></tr>
<tr><td>darCantidadMonedas1000</td><td>La cantidad de monedas de 1000.</td></tr>
<tr><td>darCantidadTotalMonedas</td><td>La cantidad total de monedas del monto.</td></tr>
<tr><td>darValorTotal</td><td>El valor total del monto.</td></tr>
<tr><td>agregarMoneda</td><td>True si pudo agregar la moneda, falso si no pudo agregarla porque sobrepasaba el valor máximo.</td></tr>
</table>
</td></tr>
<tr><td>Interrelaciones:</td><td>Ninguna</td></tr>
<tr><td>Diccionario de datos</td><td>Variables

<table>
<tr><th>Nombre de Variable</th><th>Tipo</th><th>Descripción</th></tr>
<tr><td>cantidadMonedas50</td><td>Int</td><td>Cantidad de monedas de $50 COP que hay en el monto.</td></tr>
<tr><td>cantidadMonedas100</td><td>Int</td><td>Cantidad de monedas de $100 COP que hay en el monto.</td></tr>
<tr><td>cantidadMonedas200</td><td>Int</td><td>Cantidad de monedas de $200 COP que hay en el monto.</td></tr>
<tr><td>cantidadMonedas500</td><td>Int</td><td>Cantidad de monedas de $500 COP que hay en el monto.</td></tr>
</table>
</td></tr>
</table>

cantidadMonedas1000	Int	Cantidad de monedas de $1000 COP que hay en el monto.

Constantes

Nombre de la Constante	Tipo	Descripción
VALOR_MAXIMO	Double	Valor máximo que puede tener el monto.
MONEDA_50	Int	Representa una moneda de $50 COP.
MONEDA_100	Int	Representa una moneda de $100 COP.
MONEDA_200	Int	Representa una moneda de $200 COP.
MONEDA_500	Int	Representa una moneda de $500 COP.
MONEDA_1000	Int	Representa una moneda de $1000 COP.

Métodos

Nombre de Método o función	Detalle
darCantidadMonedas50	Retorna la cantidad de monedas de 50.
darCantidadMonedas100	Retorna la cantidad de monedas de 100.
darCantidadMonedas200	Retorna la cantidad de monedas de 200.
darCantidadMonedas500	Retorna la cantidad de monedas de 500.
darCantidadMonedas1000	Retorna la cantidad de monedas de 1000.
darCantidadTotalMonedas	Retorna la cantidad total de monedas del monto.

	darValorTotal	Retorna el valor total del monto
	agregarMoneda	-Agrega una moneda al monto. -Se aumenta en uno la cantidad de monedas de la denominación dada.
	cambiarValor	-Asigna la cantidad de monedas de cada denominación necesaria para completar el valor que entra por parámetro. -Se debe garantizar el uso de la mínima cantidad de monedas. -Se asigna la cantidad de monedas requeridas a cada denominación.
	reiniciar	Reinicia todas las cantidades de las monedas. Se cambiaron a 0 todas las cantidades de monedas.
Conexiones a bases de datos:	Ninguna	
Diseño del Componente:	SI (Ver Diagrama de Clases)	
Designado a:	Juan Pablo Botina	
Rol que cumple:	Desarrollador	
Zona Responsable:	PAS-04	
Proceso:	PR-01	
Nivel de Prioridad:	Alta	
Tiempo Estimado:	8 días.	
Fecha de Inicio:	07- 04- 2018	
Fecha estimada de entrega:	13- 04- 2018	
Observaciones:	Ninguna	
Contrariedades:	Ninguna	

Definido por:	Marcela Guerrero

Tabla B 6. Plantilla Componente Interfaz Gráfica

Componentes	
Id:	COM-04
Nombre:	Componente Interfaz Gráfica Máquina Expendedora
Descripción:	Interfaz Gráfica del componente.
Características:	Ventana Interfaz Gráfica que simula la máquina expendedora.
Entradas:	- Inicializa la interfaz grafica - Recibe un componente de clase de Máquina Expendedora.
	Métodos

Nombre del Método o función	Parámetros de entrada
agregarMoneda	Denominación de la moneda.
comprarProducto	Identificador del producto. pIdentificador != null.

Salidas Esperadas:	Interfaz Gráfica Constituida.
Interrelaciones:	Componente Máquina Expendedora
Diccionario de datos	**Variables a considerar**

Nombre de Variable	Tipo	Descripción
pMoneda	Int	Denominación de la moneda.
pIdentificador	String	Identificador del producto. pIdentificador != null.

Métodos

Nombre de Método o función	Detalle
agregarMoneda	Agrega una moneda de la denominación recibida por parámetro a la

		máquina.
	comprarProducto	Realiza la compra de un producto con el identificador dado. Recibe El identificador recibido existe. Y luego Se realiza la compra y se actualiza la interfaz.
	terminarCompra	Termina la compra actual y muestra el dinero del cambio
	darValorTotalCompras	Muestra el diálogo con el valor total de compras de la máquina.
	darPorcentajeDeDisponibilidad	Muestra el dialogo con la capacidad disponible de la máquina.
	mostrarDialogoUnidadesCompradas	Muestra el dialogo con las compras por producto.
	mostrarDialogoInfoDonacionFopre	Muestra el dialogo con la donación total al FOPRE.
	darProductoMasComprado	Muestra el dialogo con el producto más comprado.
	actualizar	Actualiza la información de los paneles.
Conexiones a bases de datos:	Ninguna	
Diseño del Componente:	SI (Diseño de Interfaz Gráfica)	
Designado a:	Luis España	

Rol que cumple:	Jefe de Zona
Zona Responsable:	PAS-03
Proceso:	PR-01
Nivel de Prioridad:	Alta
Tiempo Estimado:	8 días
Fecha de Inicio:	07 – 04 – 20118
Fecha estimada de entrega:	13 – 04 – 2018
Observaciones:	Ninguna
Contrariedades:	Ninguna
Definido por:	Marcela Guerrero

Tabla B 7. Estándar de codificación

Estándar de *prácticas de* codificación	
Recomendaciones Generales.	- Escribir cada componente de software en español incluyendo comentarios. - Usar paquetes o diccionarios para mostrar la información al usuario final en el lenguaje nativo. - Las líneas en blanco o comentarios no cuentan como líneas de código. - Separar en archivos la parte de lógica de negocios con la capa de presentación.
Formato de cabecera	Describe como debe organizar la información de cabecera en cada archivo que se realice con el fin de que cada colaborador lo tenga en cuenta al momento de codificar. *@nombre:* Nombre del componente. *@objetivo:* Detalla brevemente el objetivo general del componente. *@descripción*. Detalla brevemente lo que hace el componente. *@colaborador:* Nombre del colaborador que realizó el componente. *@zona:* Identificador de la zona a la cual pertenece. *@fecha:* Fecha de entrega del componente. *@versión:* Número de entrega correspondiente al componente. (Incrementos de manera decimal)
Ejemplo Formato de cabecera.	*@nombre:* Componente para el registro de usuarios a la plataforma. *@objetivo:* Desarrollar un componente apto para facilitar el registro de usuarios al sistema de notas de la Universidad de Nariño. *@descripción:* Componente para el registro de usuarios al sistema sea docentes tiempo completo u hora catedra de la Universidad de Nariño, solicitando la información básica del usuario. *@colaborador:* Nelson Suarez *@zona:* PAS-03 *@fecha:* 10 – febrero 2017 *@versión.* 1.0

<table>
<tr><td rowspan="6">Organización de Variables.</td><td>Especifica la manera en cómo las variables que puedan ser usadas, debe ser declaradas siguiendo un orden en específico y además escritas una por cada sentencia.</td></tr>
<tr><td>Ejemplo:
Correcto:
int contador = 1;
string saludo = “bye”;

Incorrecto:
string saludo = “bye”; int contador = 1;</td></tr>
<tr><td>Organización de Variables.</td><td>- Ubicación de variables primero las de clase public, luego las protected y finalmente las private.
- Organización de acuerdo al tipo de dato (int, float, double), tipo carácter, tipo texto, tipo booleano, arreglos o matrices.</td></tr>
<tr><td>Ejemplo:
Correcto:
public int contador = 0;
public string nombre = “Maria”;
public boolean bandera = true;
public string idEstudiante = ‘212’;

private int suma = 0;
private string teléfono = “8383”;

Incorrecto:
private string saludo = “hola”;
public int numero = 2;</td></tr>
<tr><td>- Nombre de variables compuesta con el formato LowerCamelCase.</td></tr>
<tr><td>Ejemplo:
Correcto de la Forma LowerCamelCase:
int contarLlegadas = 0;
Incorrecto
int countllegadas = 0;</td></tr>
</table>

	-No permita que el nombre de las variables sea la combinación de más de dos idiomas.

	Ejemplo:
	Correcto:
	`int contarSalidas = 0;`
	
	Incorrecto:
	`int countSalidas = 0;`

Inicialización de Variables	Procure inicializar las variables que va a usar al momento de su declaración.
	Ejemplo:
	Correcto
	`public int dinero = 1000;`
	`private String[] profesion = {"Sistemas", “Administración", "Psicologia" };`

Constantes	- Se sugiere que el nombre de las constantes se escriba en mayúscula.
	Ejemplo:
	Correcto:
	`int AREA_MAXIMA;`
	`int EDAD_MINIMA;`

Indentación	Propiciar que el código este ordenado en base a las estructuras que se vayan planteando. Se propone una Indentación común de 4 espacios equivalente al uso del tabulador.

Operadores	Permite establecer la declaración de operadores dentro del código fuente.
	Agregar espacios entre:
	- Operandos de declaración de variables.
	Ejemplo:
	`int edad = 5;`
	
	- Entre valores de operaciones aritméticas
	Ejemplo:
	`m = 5 + 2;`
	
	- Entre operadores lógicos o de comparación como `<=, >=, <, >, !=, &&, ||.`

Operadores	Ejemplo:
	`if(variable > 4 && variable < 10){`
	`  cout<<“Correcto”;`
	`}`
	
	Use signos de agrupación para separar los operadores lógicos cuando se usan más de uno en una misma instrucción.
	
	Ejemplo:
	`if((edad > 15 && edad < 20) || (genero == ‘F’)){`
	`  alert(“comprobado”);`
	`}`

Comentarios	- Los comentarios pueden ser de una línea y de múltiples líneas. - Después de un comentario aplique un salto de línea. - Los comentarios deben ir con lo estrictamente necesario. - Los comentarios deben mantener que no ocupen más de 80 caracteres por línea.

Funciones, métodos internos o clases	-Definir las funciones internas que pueda contener el componente software con nombres pertinentes a lo que se está realizando y que no superen los 30 caracteres.
	-Evitar colocar espacios entre el nombre de un método o función y el paréntesis de apertura de esta.
	Ejemplo:
	Correcto:
	`public void verEstudiantes()`
	
	- Definir cada método o función empleada con el formato lowerCamelCase.
	Ejemplo:
	Correcto:
	`void consultarRegistro()`
	
	- Coloque espacios en un método o función entre cada coma que separa un parámetro de otro.
	Ejemplo:
	`void consultarNotas(a, b, c);`

Funciones, métodos internos o clases	- Separe cada método, función o clase con una línea en blanco. -Las funciones o métodos deberán nombrarse de acuerdo al listado de palabras ofrecidas en el diccionario de datos de cada componente.

División de líneas.	Cuando una expresión no alcance por su tamaño en una sola línea esta podrá continuar en la siguiente siempre y cuando:
	- Exista una coma
	- Exista un operador.
	
	Ejemplo:
	`int recibirCliente(nombre, apellido, edad, teléfono);`
	
	`if ((condicion1 && condicion2)`
	`|| (condicion3 && condicion4)`
	`|| (condicion5 && condicion6)) {`
	`recibirCliente('maria','gomez',`
	`31,'722881');`
	`}`

Reutilización de código	Para evitar que exista código repetido dentro del desarrollo del componente, haga uso de la reutilización de código fuente mediante el uso de funciones, economice tiempo y trabajo doble.

Tabla B 8. Estándar para plataformas de modelado, entornos de programación y gestión de código fuente

Plantilla de entornos de programación	
Recomendaciones Generales.	A continuación, se listan las herramientas que va usar para trabajar en el desarrollo de software del componente.
Lenguajes de programación.	`Java`
Frameworks para su uso.	`Ninguno`
Entornos de desarrollo	`Netbeans`
Editor de texto	`Sublime Text`
Gestor de base de datos	`NA`
Gestor de librerías o dependencias.	`Github`
Modelador de Software	`StarUML`

Tabla B 9. Estándar para herramientas de comunicación y gestión de trabajo

Plantilla de herramientas de gestión de trabajo	
Recomendaciones Generales.	A continuación, se listan las herramientas que va usar para la gestión de trabajo y comunicación en el proyecto de desarrollo de software.
Plataformas para la comunicación.	
Descripción general	Herramientas de comunicación que van a ser empleadas dentro del proyecto como parte fundamental para la interacción entre los participantes.
Herramientas a usar.	**Skype.**
	Correo electrónico.
Plataforma para la administración de proyectos.	
Descripción general	Orientada al seguimiento de actividades en línea, de cada una de las tareas que cada colaborador esta efectuado y el estado real de ellas.
Plataforma a usar	**Trello.**

Tabla B 10. Estándar para la documentación del Sistema

Plantilla para la documentación del Sistema	
Arquitectura	Aplicación Java
Fuente	**Para líneas de código:** **Tipo de Fuente**: Consolas
Modelado	Cada diagrama, modelado o diseño que se haga del componente deberá tener un título y si se desea una descripción sencilla (no supere las 20 palabras).

Funciones o métodos	Antes de iniciar una función, clase, método o procedimiento según sea el caso haga una breve descripción de lo que esta realiza.
	Ejemplo
	/**
	* Representa un producto de la máquina expendedora.
	*/
	public class Producto
	{
	
	}

Interfaces	Interfaz en Java - Utilice herramientas Java para la realización de la interfaz. - Utilice como apoyo el diseño propuesto

Glosario de Herramientas	**Correo electrónico**. De carácter formal para él envió de las plantillas de funcionalidades, procesos y componente entre los participantes, así como la asignación de actividades, tutoriales y estándares establecidos. **Github**. Es una plataforma para desarrollo de software colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de computadora. **Netbeans.** Es un entorno de desarrollo gratuito y de código abierto. Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Da soporte a tecnologías como Java, PHP, Groovy, C/C++, HTML5, entre otras. **StarUML.** Es una herramienta para el modelamiento de software basado en los estándares UML (Unified Modeling Language) y MDA (Model Driven Arquitecture). Permite crear Diagrama de casos de uso, Diagrama de clase, de secuencia, de colaboración, de estados, de actividad, Diagrama de componentes, de despliegue y de composición estructural. **Skype.** Empleada para la realización de reuniones grupales de manera virtual cara a cara, entre colaboradores y jefes de zona. **Trello.** De uso diario para la administrar el flujo del proyecto en su totalidad y de los procesos asignados por zona para tener un control de los avances con mayor facilidad, organizado y midiendo su progreso.
Tutorial de Herramientas.	Trello. https://www.youtube.com/watch?v=tOpBJnOifAc Skype https://www.youtube.com/watch?v=4gjnviTZzew Github https://www.youtube.com/watch?v=mlt1t5zDtbs

Tabla B 11. Plantilla Información del Equipo de trabajo

Plantilla Información de Zona		
Id:	PA-3	
Descripción:	Zona conformada por estudiantes desarrolladores de la Universidad de Nariño del programa de Ingeniería de Sistemas	
Jefe de Zona:	Luis España	
Número de Participantes:	3	
Presenta Desfase horario:	No	
Horas de diferencia:	NA	
Colaboradores:		
Nombre		**Ubicación Geográfica**
Alexis Camilo Valencia		Pasto – Nariño
Juan Pablo Botina		Pasto – Nariño
Daniel Tutistar		Pasto – Nariño

ANEXO C. SEGUIMIENTO DEL TRABAJO - CASO PRÁCTICO

Figura C. 1 Uso de Trello, Primera Fase de trabajo

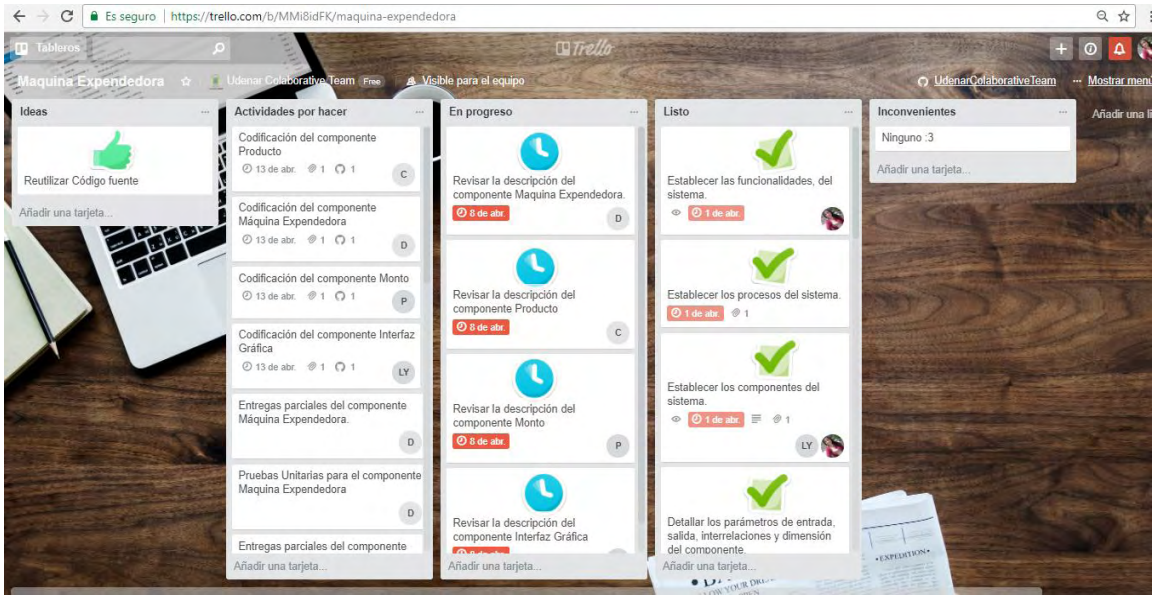


Figura C. 2 Uso de Trello, Seguimiento de Actividades

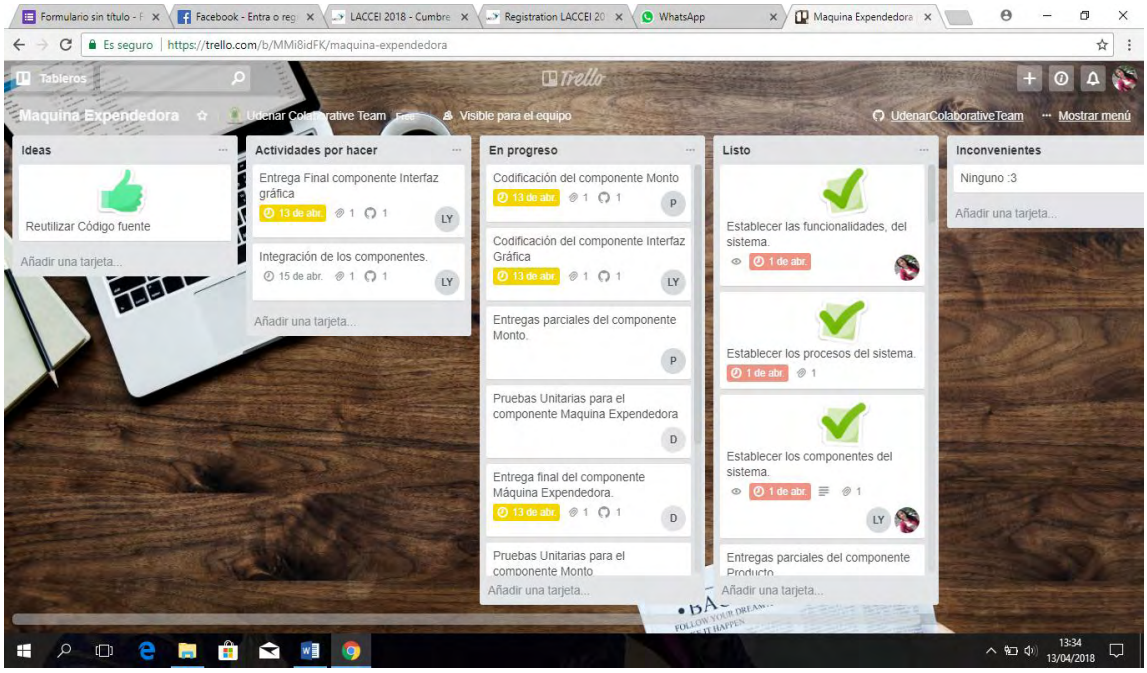


Figura C. 3. Uso de Trello, Finalización del trabajo

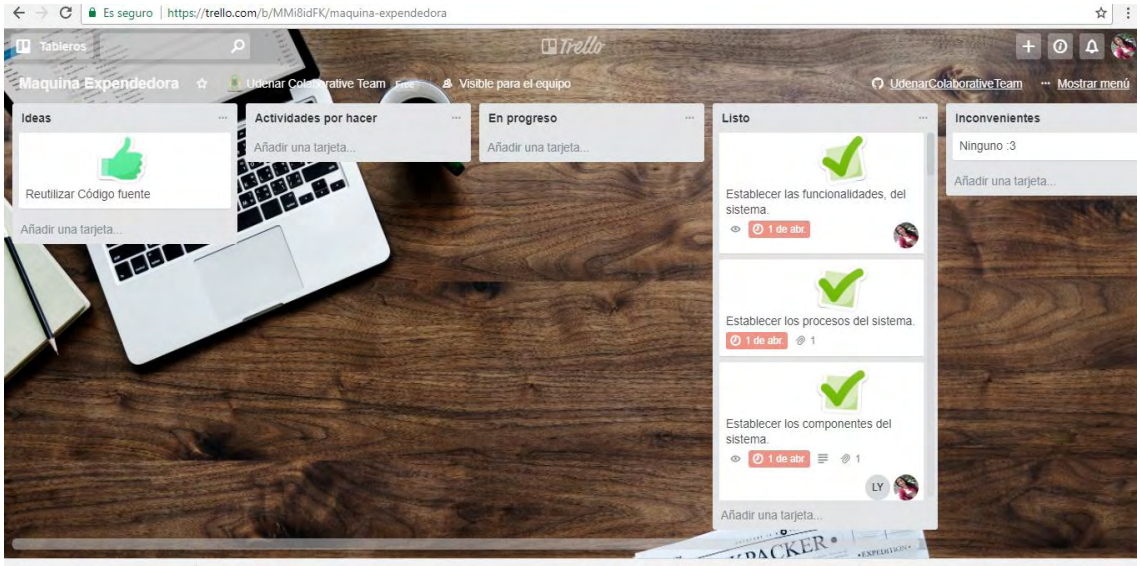


Figura C. 4. Uso de Github, Creación del Repositorio.

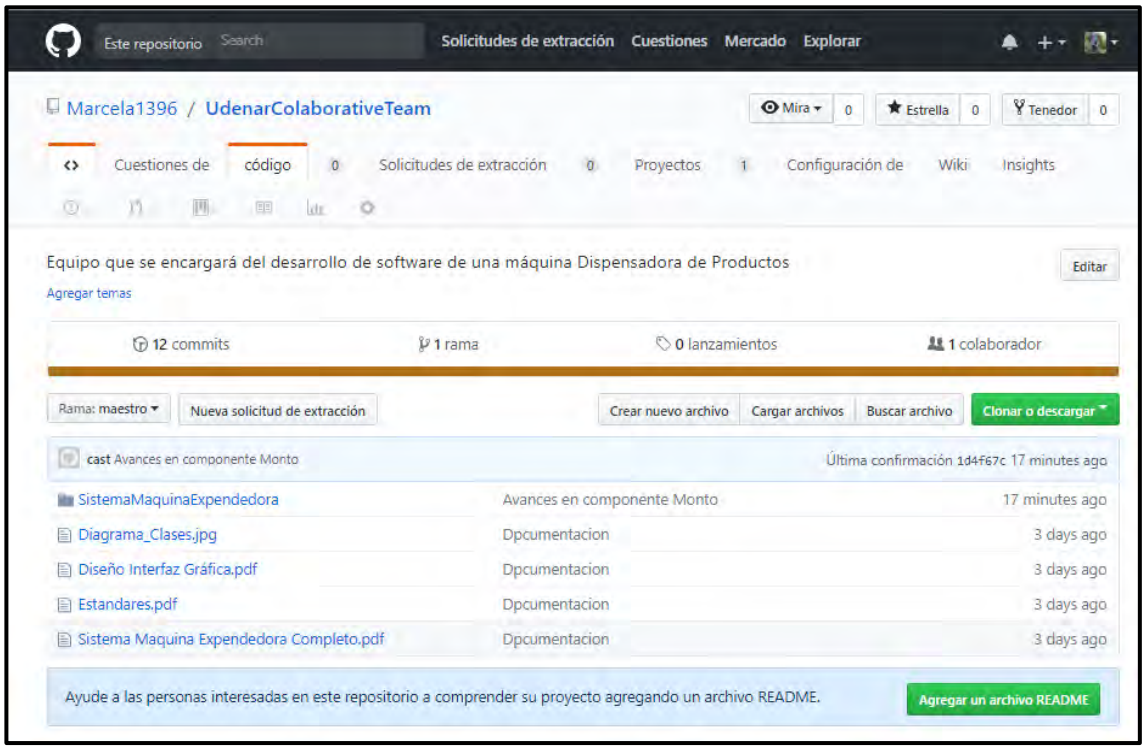


Figura C. 5. Uso de Github, Contribuciones de los colaboradores

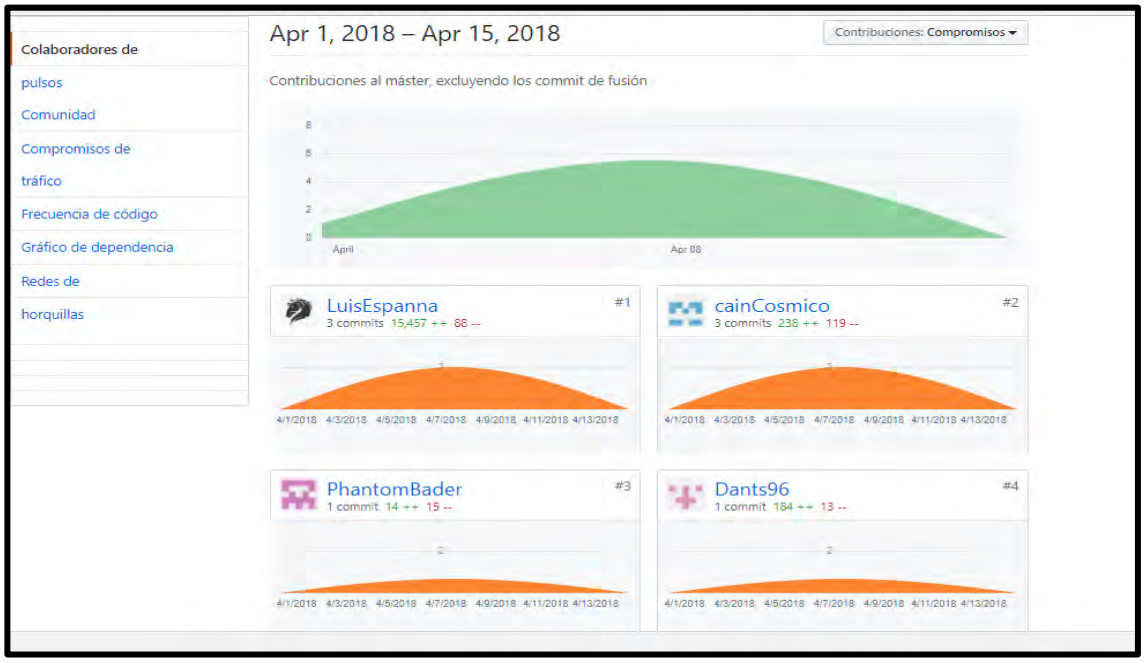


Figura C. 6. Uso de Github, Proyecto Culminado

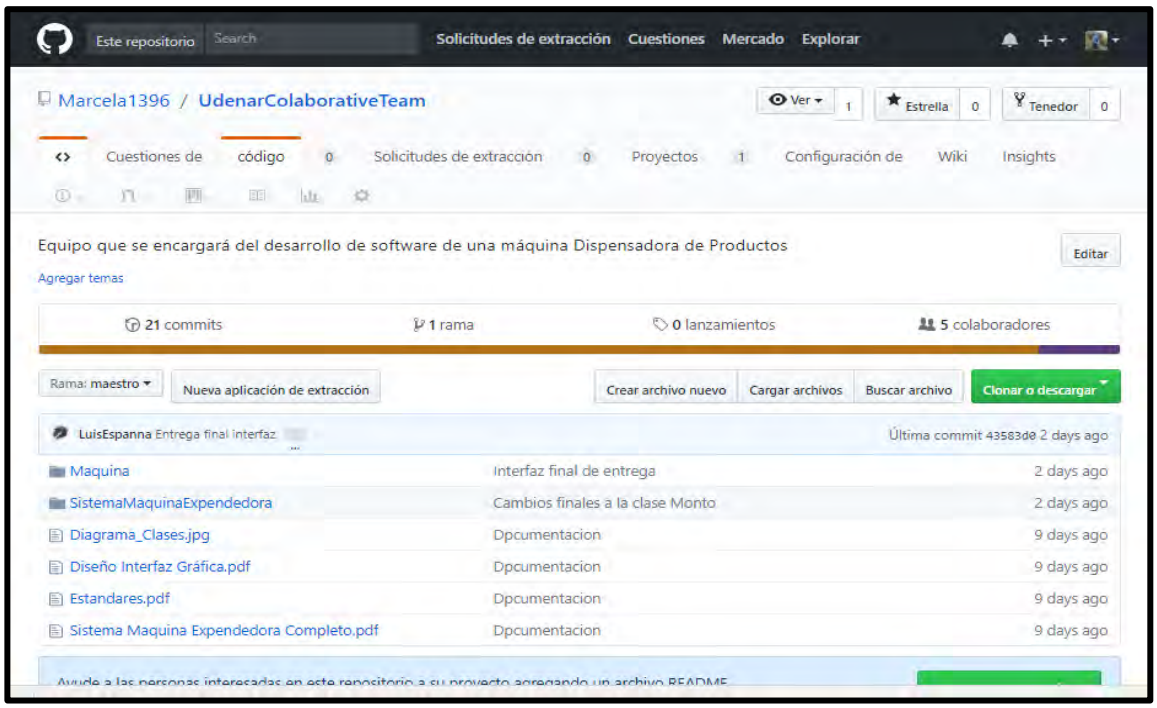


Figura C. 7. Presentación Componente Máquina

Rama: maestro ▾ UdenarColaborativeTeam / Maquina / src / clases / MaquinaExpendedora.java Encontrar archivo Copiar ruta

LuisEspanna Entrega final interfaz 43583d8 2 days ago

1 colaborador

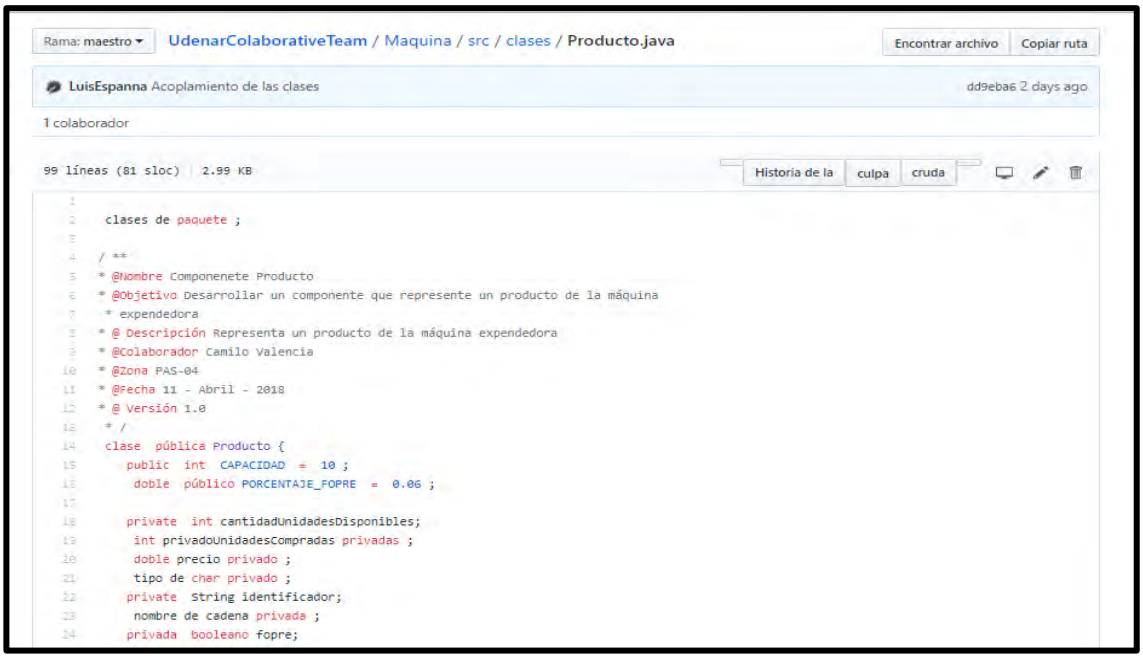
176 líneas (137 sloc) 5.39 KB Historia de la culpa cruda

```
clases de paquete ;

/ **
* @nombre: Componente Máquina Expendedora
* @objetivo: Construir una aplicación que permita simular
 * el uso, por parte de un cliente, de una máquina
 * expendedora de alimentos, la cual contiene 4
* productos. La aplicación permite al usuario (o cliente
 * de la máquina) añadir monedas a la máquina para
 * tener un crédito de compra, comprar productos, recibir
 * el cambio cuando haya terminado una compra y
 * conocer la información sobre las compras que ha
 * realizado en la máquina.
* @ descripción: La máquina expendedora cuenta con cuatro productos de los
 * cuales pueden ser comprados por un usuario de acuerdo al dinero que este
 * ingrese.
* @colaborador: Daniel Tutistar
* @zona: PAS-04
* @fecha: 10 - Abril 2018
* @ versión. 1.0
* /

clase pública MaquinaExpendedora {
```

Figura C. 8. Presentación Componente Producto



Rama: maestro ▾ UdenarColaborativeTeam / Maquina / src / clases / Producto.java Encontrar archivo Copiar ruta

LuisEspanna Acoplamiento de las clases dd9eba6 2 days ago

1 colaborador

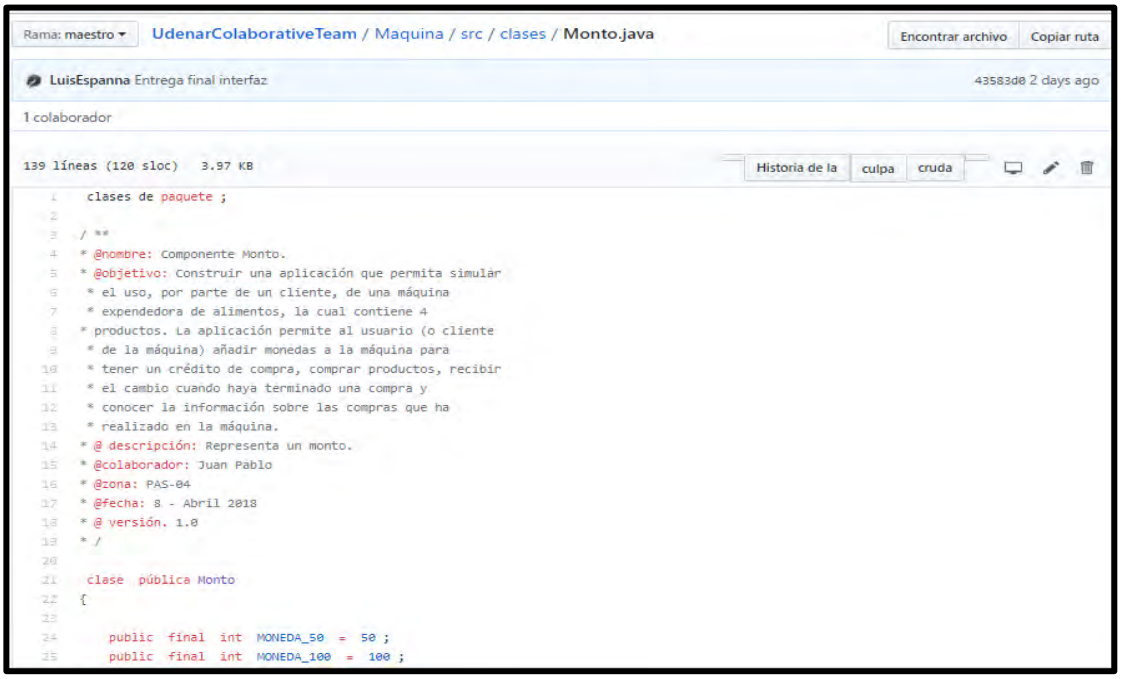
99 líneas (81 sloc) 2.99 KB Historia de la culpa cruda

```
clases de paquete ;

/ **
* @Nombre Componenete Producto
* @Objetivo Desarrollar un componente que represente un producto de la máquina
 * expendedora
* @ Descripción Representa un producto de la máquina expendedora
* @Colaborador Camilo Valencia
* @Zona PAS-04
* @Fecha 11 - Abril - 2018
* @ Versión 1.0
 * /
clase pública Producto {
   public int CAPACIDAD = 10 ;
    doble público PORCENTAJE_FOPRE = 0.06 ;

   private int cantidadUnidadesDisponibles;
    int privadoUnidadesCompradas privadas ;
    doble precio privado ;
    tipo de char privado ;
   private String identificador;
    nombre de cadena privada ;
   privada booleano fopre;
```

Figura C. 9. Presentación Componente Monto



Rama: maestro ▾ UdenarColaborativeTeam / Maquina / src / clases / Monto.java

Encontrar archivo Copiar ruta

LuisEspanna Entrega final interfaz 43583d8 2 days ago

1 colaborador

139 líneas (120 sloc) 3.97 KB

Historia de la culpa cruda

```
clases de paquete ;

/ **
* @nombre: Componente Monto.
* @objetivo: Construir una aplicación que permita simular
 * el uso, por parte de un cliente, de una máquina
 * expendedora de alimentos, la cual contiene 4
* productos. La aplicación permite al usuario (o cliente
 * de la máquina) añadir monedas a la máquina para
 * tener un crédito de compra, comprar productos, recibir
 * el cambio cuando haya terminado una compra y
 * conocer la información sobre las compras que ha
 * realizado en la máquina.
* @ descripción: Representa un monto.
* @colaborador: Juan Pablo
* @zona: PAS-04
* @fecha: 8 - Abril 2018
* @ versión. 1.0
* /

 clase  pública Monto
{

    public  final  int  MONEDA_50  =  50 ;
    public  final  int  MONEDA_100  =  100 ;
```

ANEXO D. DISEÑO DEL SISTEMA - CASO PRÁCTICO

Figura D. 1 Diagrama de Clases, Caso Práctico

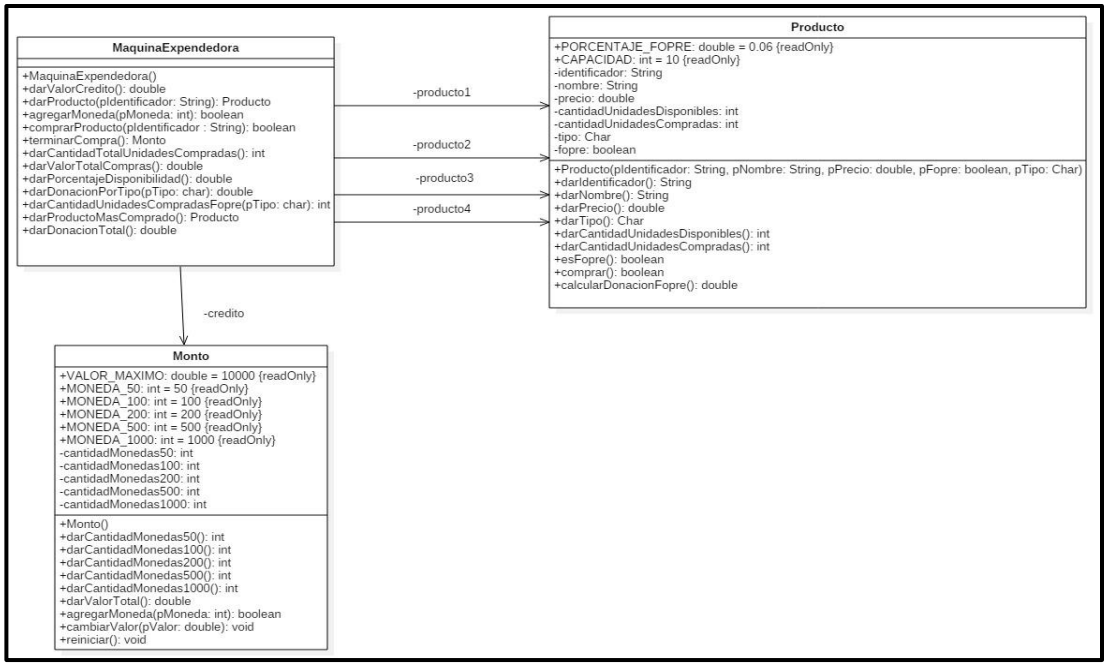


Figura D. 2. Interfaz Gráfica Parte 1, Caso Práctico

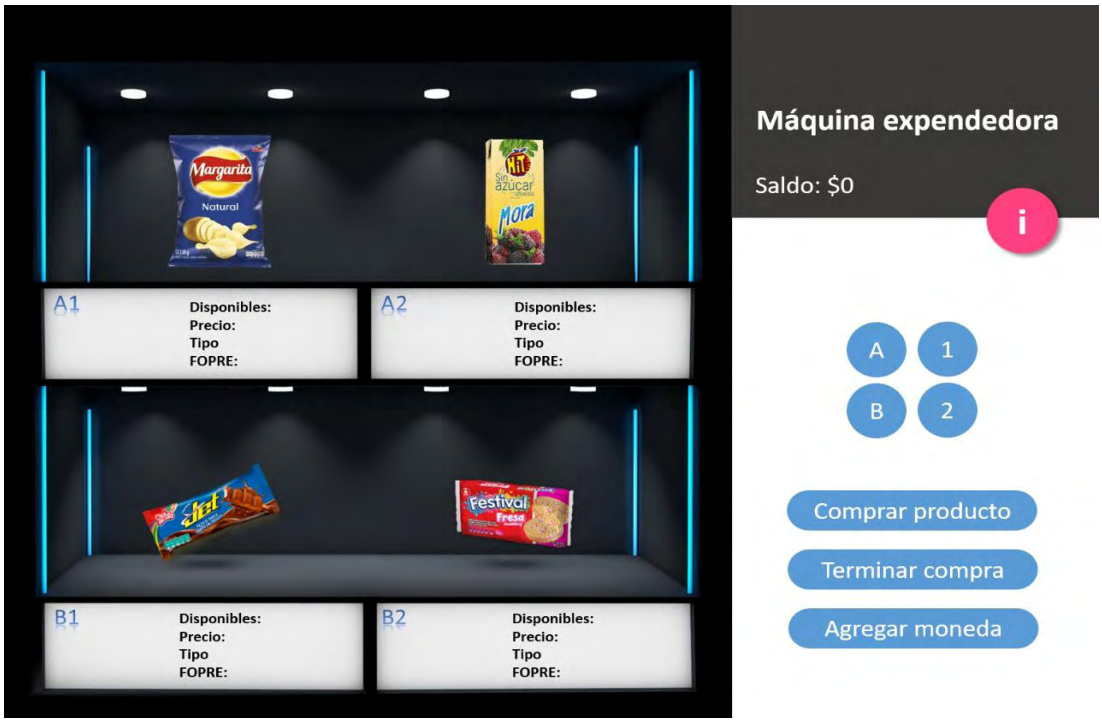


Figura D. 3. Interfaz Gráfica Parte 2 - Caso Práctico

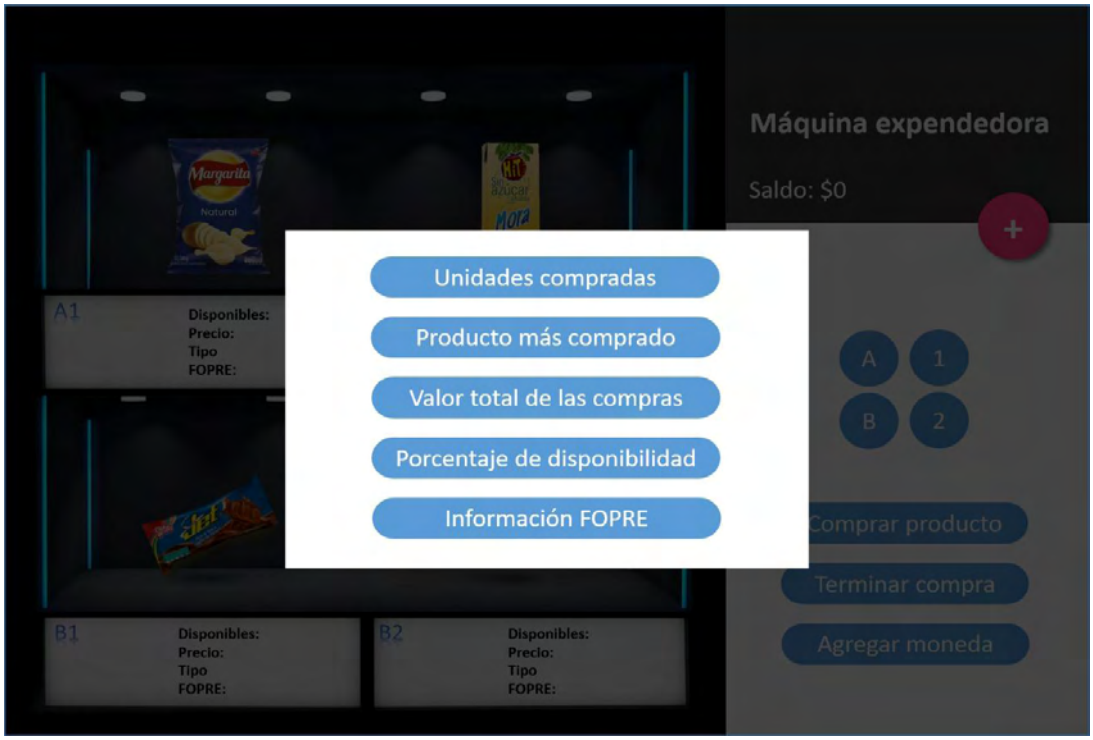


Figura D. 4 Interfaz Gráfica Parte 3 - Caso Práctico