# METODOLOGÍA ADAPTATIVA BASADA EN SCRUM, APOYADA POR HERRAMIENTAS DE SOFTWARE LIBRE PARA EL CENTRO DE INFORMÁTICA DE LA UNIVERSIDAD DE NARIÑO

**GEOVANY STEVEN VITERI SALAZAR**

**UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA**
**FACULTAD DE INGENIERÍA DE SISTEMAS**
**PROGRAMA DE MAESTRÍA EN SOFTWARE LIBRE CONVENIO UNAB-UOC**
**SAN JUAN DE PASTO**
**2017**

# METODOLOGÍA ADAPTATIVA BASADA EN SCRUM, APOYADA POR HERRAMIENTAS DE SOFTWARE LIBRE PARA EL CENTRO DE INFORMÁTICA DE LA UNIVERSIDAD DE NARIÑO

**GEOVANY STEVEN VITERI SALAZAR**

**Trabajo de grado como requisito para optar al título de Magister en Software Libre**

**Director GIOVANNI ALBEIRO HERNÁNDEZ PANTOJA**

**Magister en Docencia Universitaria**

**UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA**
**FACULTAD DE INGENIERÍA DE SISTEMAS**
**PROGRAMA DE MAESTRÍA EN SOFTWARE LIBRE CONVENIO UNAB-UOC**
**SAN JUAN DE PASTO**
**2017**

**NOTA DE ACEPTACIÓN:**

_____
_____
_____
_____
_____
_____
_____

_____
Firma Presidente del jurado

_____
Firma del Jurado

_____
Firma del Jurado

San Juan de Pasto, 30 de Agosto de 2017.

## DEDICATORIA

*Dedico el presente trabajo a papito Dios, que siempre abrió las puertas y me dio la fuerza, sabiduría y energía para culminar satisfactoriamente cada una de las fases de este proceso. A mi madre (QEPD) por ser mi modelo a seguir con su amor, fuerza y valentía. A mi padre por inculcarme entrega y perseverancia. A todas mis hermanas por su apoyo constante y gran amor y a todos aquellos que participaron directa o indirectamente en la elaboración de este trabajo.*

***Geovany Steven Viteri Salazar***

## AGRADECIMIENTOS

A mi asesor el M, Sc. Giovanni Albeiro Hernández Pantoja, por su orientación y apoyo para la realización de este trabajo.

Al Centro de Informática de la Universidad de Nariño.

A la iniciativa Talento TI, del Ministerio de Tecnologías de la información y las comunicaciones.

## RESUMEN

El presente proyecto de investigación consolida una metodología para el soporte, mantenimiento y construcción de software, basada en Scrum y apoyada por herramientas de software libre para el Centro de Informática de la Universidad de Nariño. Para lograrlo, se caracteriza y compara los elementos metodológicos del proceso de construcción de software del Centro de Informática y los definidos por Scrum. Luego, se formula una propuesta basada en Scrum, apoyada por herramientas de software libre; finalmente, se aplica en el desarrollo de un producto software. Está investigación, se enmarca dentro del paradigma cuantitativo, con enfoque empírico-analítico de tipo descriptivo y propositivo. La población objeto de estudio fueron las Universidades públicas del sur-occidente colombiano, con muestreo no probabilístico de tipo intencional, seleccionando a la Universidad de Nariño. Como resultado se logró caracterizar y comparar los elementos metodológicos de Scrum y los del Centro de Informática, con base en este análisis se plantea una propuesta de trabajo en equipo, basada en Scrum y Peopleware, compuesta por cuatro adopciones incrementales, en las se incorporaron los elementos metodológicos etapa, rol, herramienta, artefacto y lineamiento propuesto en Scrum, y las técnicas Niko Niko, Pomodoro, Kanban, MobProgramming y Coding Dojo. La propuesta se apoyó por herramientas de software libre, en donde se evaluaron varias utilizando un modelo abierto, la herramienta Kunagi obtuvo el mayor ponderado final, por consiguiente fue la seleccionada. Finalmente, se aplica la propuesta en el desarrollo de un producto software. El trabajo permite concluir que en el Centro de Informática de la Universidad de Nariño no se logra evidenciar que exista una forma o método estandarizado y explicito que propicie el trabajo en equipo. Las principales similitudes entre el proceso de software del Centro de Informática de la Universidad de Nariño y Scrum obedecen a aspectos relacionados con la pila de producto, el *done* y el *development team*. La diferencia más relevante corresponde a la ausencia de métricas.



### Palabras Clave

Metodología Ágil, *Peopleware*, Scrum, Software Libre.

## ABSTRACT

The present research project consolidates a methodology for the support, maintenance and construction of software, Based on Scrum and supported by free software tools for the Computing Center of the University of Nariño. To achieve this, it is characterized and compared the methodological elements the software construction process of the Computing Center and those defined by Scrum. A proposal based on Scrum is formulated, supported by free software tools; finally, is applied in the development of a software product. This research is within the quantitative paradigm, with empirical-analytical approach of descriptive and propositional type. The study population was the public universities of south-west of Colombia, with non-probabilistic sampling of the intentional type, selecting the University of Nariño. As a result it was possible to characterize and compare the methodological elements of Scrum and those of the Computer Center, based on this analysis it propose a teamwork proposal, based on Scrum and Peopleware, composed of four incremental adoptions, incorporating the methodological elements stage, role, tool, artifact and guideline proposed in Scrum, and Niko Niko, Pomodoro, Kanban, MobProgramming and Coding Dojo techniques. The proposal was supported by free software tools, where several were evaluated using an open model, the Kunagi tool obtained the highest final weighting, therefore it was selected. Finally, the proposal is applied in the development of a software product. The work allows concludes that in the Computing Center of the University of Nariño it is not possible to demonstrate that there is a standardized and explicit shape or method that favors teamwork. The main similarities between the software process of the Computer Center of the University of Nariño and Scrum obey aspects related to the product stack, the *done* and the development team. The most relevant difference corresponds to the absence of metrics.



### Keywords

Agile Methodology, Free Software, *Peopleware*, Scrum.

**CONTENIDO**

Pág.

## LISTA DE TABLAS

Pág.

## LISTA DE FIGURAS

Pág.

## LISTA DE ANEXOS

Pág.

# INTRODUCCIÓN

Actualmente, el proceso de creación de software es una tarea compleja, ya que depende de las características, tamaño del proyecto y el contexto en el que se desarrolla, en donde a pesar de que se cuentan con numerosas herramientas y metodologías, de nada sirven si no se proveen las directivas necesarias que las implementen y ajusten sus lineamientos a la forma particular de trabajo, con el objetivo de lograr una sinergia y mejorar la calidad del producto y los tiempos de entrega. Estos retos que se plantea en el soporte, mantenimiento y construcción de software, no son ajenos a las Instituciones de Educación Superior, en donde todos los procesos académicos y administrativos deben ser soportados por productos software, que deben cumplir estándares de calidad, de accesibilidad, interoperabilidad y usabilidad. En el caso de la Universidad de Nariño, para dar respuesta a estos requerimientos existe un departamento de tecnología denominado Centro de Informática, quien es consciente de que un factor que aporta al aseguramiento de la calidad del software es la forma de trabajo, para hacer frente a la complejidad y necesidades crecientes de software en la Universidad, las cuales demandan recursos técnicos y humanos que muchas veces están por debajo de lo ideal; en donde cada funcionario es responsable de un proyecto en particular y en caso de renuncia a veces la vacante no se cubre, y las funciones y responsabilidades son transferidas al personal existente, generando sobrecarga laboral y estancamientos y retrasos en la producción de software. Cuando se da el caso contrario, en el cual hay una nueva contratación, debido a que no existen unos procedimientos claros y definidos sobre inducción y existencia de manuales de usuario y programador actualizados, el nuevo personal tiene que adquirir el conocimiento de lo que este a su cargo sobre la marcha, analizando el código fuente para identificar su funcionalidad; lo cual trae como consecuencia, una prolongación en la curva de aprendizaje del funcionario para iniciar a ser productivo en el CI. En cuanto a la etapa de pruebas para el software desarrollado, esta se simplifica a la prueba-error, donde el mismo programador y el cliente son quienes detectan los defectos, en ocasiones cuando el producto ya es publicado y está en producción, sin llevar un registro de los mismos y el tratamiento que se les ha dado para solucionarlos.

Lo anterior se presenta porque la forma de trabajo de los funcionarios del CI no está claramente estructurada y se realiza en gran parte de manera individual, sin seguir unos lineamientos definidos, en donde un buen desempeño obedece a la experiencia y conocimiento de quién esté realizando el soporte, mantenimiento o nuevo desarrollo, derivando en una alta carga laboral para los ingenieros, que impide cumplir con todos los requerimientos de usuario, que por lo general siempre son cambiantes y urgentes, a veces renunciando a prácticas esenciales que aseguran la calidad del producto.

Para dar solución al problema descrito anteriormente, en este trabajo investigativo se tiene como fin consolidar una metodología para el soporte, mantenimiento y construcción de software, basada en Scrum y apoyada por herramientas de software libre; es decir, destacar las ventajas que ofrece los lineamientos definidos en el marco de trabajo de Scrum y como obtener sus beneficios. Para alcanzar este propósito, en primera instancia, se realizará una caracterización de los elementos metodológicos del proceso de construcción de software del CI y los de Scrum, con el fin de obtener una visión clara y real de la situación actual. Luego, se desarrollará una comparación de los elementos metodológicos mencionados anteriormente, para establecer el nivel de similitud en el cual se encuentran estas dos prácticas, y de esta manera poder formular una propuesta basada en este marco de trabajo. Finalmente, esta propuesta metodológica, se aplicará en el proceso de construcción de un producto software, que permita evaluar el nivel de aporte al CI.

Para Canos, Letelier y otros autores (2003), la agilidad en un proceso, se presenta cuando el desarrollo del software es incremental, cooperativo, sencillo y adaptable; características que permiten obtener un resultado de forma efectiva y eficiente. Por lo que una empresa u organización que desarrolla sus actividades con fundamento en este principio, pueda aplicar ese tipo de agilidad y tener más éxito en su negocio.

Es importante considerar que los proyectos que no tienen muy definidos sus requisitos, o que la estabilidad en ellos es muy frágil, se acoplan perfectamente a las metodologías agiles, ya que la rápida respuesta a cambios permite que el producto sea adaptable y los riesgos sean mínimos y de bajo impacto, lo cual se acopla perfectamente al contexto del CI.

Uno de los marcos de trabajo ágiles para la gestión de proyectos que ha sido ampliamente utilizada en los últimos años es Scrum. Schwaber y Sutherland (2013, pág. 4) manifiestan que Scrum es un conjunto de lineamientos orientados a la gestión del desarrollo de software, que busca agregar valor a los procesos de negocio de quién la aplique. Se fundamenta en enfocarse en los productos funcionales más importantes para el cliente y verificación continua, nivel de adaptación e innovación; es un proceso en donde se utilizan y aplican mejores prácticas para el trabajo colaborativo, en equipo que maximicen el resultado.

Esta investigación es de corte cuantitativo, con un enfoque empírico-analítico de tipo descriptivo. La población objeto de estudio corresponde a las Universidades públicas del Sur-Occidente Colombiano. La muestra es no probabilística de tipo intencional para el caso la Universidad de Nariño, específicamente el Centro de Informática. Las técnicas e instrumentos de recolección para la presente investigación serán la encuesta y la revisión documental, y como técnica de análisis se aplicará estadística descriptiva y análisis documental. Las variables en las cuales se basarán las métricas para el análisis serán: Etapa, Actividad, Rol, Artefacto, Herramienta y Lineamiento.

Como resultado de la investigación se obtuvo la caracterización y comparación de los elementos metodológicos de Scrum y aquellos que componen el proceso de construcción, soporte y mantenimiento de software en el Centro de Informática, con base en este análisis se plantea una propuesta de trabajo en equipo, basada en Scrum y *Peopleware*, compuesta por cuatro adopciones incrementales, en las se incorporaron los elementos metodológicos etapa, rol, herramienta, artefacto y lineamiento propuestos en Scrum, y las técnicas como Niko Niko para registrar el estado de ánimo del equipo y mejorar su motivación, Pomodoro con el fin de administrar el tiempo de manera eficiente y mejorar la productividad, Kanban para visualizar el trabajo en equipo, MobProgramming en la solución de problemas de manera colectiva y Coding Dojo que contribuyen en la socialización de conocimiento entre los integrantes del equipo. La propuesta se apoyó por herramientas de software libre, en donde se evaluaron varias utilizando un modelo abierto conformado por cuatro fases: planeación, ejecución, verificación y selección, en el que la herramienta Kunagi obtuvo el mayor ponderado final, por consiguiente fue la seleccionada. Finalmente, se aplica la propuesta en el desarrollo de un producto software por un equipo de cuatro integrantes.

# 1. ELEMENTOS DEL PROCESO INVESTIGATIVO

## 1.1 ANTECEDENTES Y ESTADO DEL CONOCIMIENTO

Tabla 1. Marco para evaluar el valor en metodología Scrum

| **Título antecedente** | **Marco para evaluar el valor en metodología Scrum (Colla, 2012)** |
|---|---|
| **OBJETIVO GENERAL** | |
| Considerar y calcular como el uso de Scrum contribuye a crear valor (Valor Presente Neto) a través de la reducción de riesgos en cuanto a resultados del proyecto, es decir, reduciendo la incertidumbre, y teniendo en cuenta el valor de las opciones que surgen a partir de la gestión. | |
| **RESULTADOS** | |
| Los resultados son medidos de acuerdo a la variación del valor presente neto y el valor aportado por las opciones del proyecto, estudiando las diferencias con organizaciones maduras teniendo como referencia sus valores como esperables. El uso de Scrum encaminado a satisfacer los objetivos de calidad de SEI-CMMI por medio del incremento de valor del proyecto crea valor en la organización. Es posible especular que existe una proporción directamente proporcional entre el número de sprints, opciones y el valor generado por ellas; pero si se incrementa sprints entonces se incrementa el costo fijo, por lo tanto se debe determinar el número óptimo de sprints en cada proyecto. | |
| **CONCLUSIONES** | |
| La aplicación de Scrum en organizaciones pequeñas y medianas da lugar al despliegue de prácticas que por su madurez (consistentes con nivel 2 y 3 del modelo SEI-CMMI) eleva el rendimiento y competitividad en todas las operaciones. La implementación de Scrum conlleva a una inversión reducida pero que permite ver un gran beneficio en su uso. | |
| **SIMILITUDES CON LA INVESTIGACIÓN** | |
| La principal semejanza con esta investigación es que utiliza la metodología Scrum para llevar a cabo un proyecto, es decir, hace uso de los lineamientos que plantea esta metodología para evaluar las ventajas y desventajas en el uso. | |
| **DIFERENCIAS CON LA INVESTIGACIÓN** | |
| La diferencia fundamental con esta investigación, se centra en que a partir de una caracterización de los elementos metodológicos del proceso de construcción de software en el centro de informática de la Universidad de Nariño, se realizará una contrastación con los de Scrum, para posteriormente formular una propuesta de trabajo propia para dicha dependencia. | |

Fuente: La presente investigación - 2017

Tabla 2. *Facilitating problem-based learning in teams with Scrum*

| **Título antecedente** | ***Facilitating problem-based learning in teams with Scrum (Ovesen, 2013)*** |
|---|---|
| **OBJETIVO GENERAL** | |
| Descubrir como la aplicación de la metodología Scrum aporta y mejora en el desarrollo de un proyecto organizado y en el aprendizaje basado en problemas dentro de dos equipos de estudiantes de la universidad de Aalborg. | |
| **RESULTADOS** | |
| La utilización de Scrum dentro de los dos casos de estudio demostró que fortaleció las prácticas de gestión permitiendo que los equipos trabajen de una forma más eficiente y enfocada, y como aporte adicional y muy importante, mejoro considerablemente la comunicación dentro de los miembros del equipo. | |
| **CONCLUSIONES** | |
| El uso de Scrum facilita ver la proyección diaria del desarrollo de un proyecto, pero dificulta ver la planificación a largo plazo en el caso que no se aplique el marco completo de Scrum, además el aprendizaje basado en problemas se mira beneficiado al aplicar Scrum en la parte de gestión de proyectos, es decir, el aprendizaje y la gestión se ayudan mutuamente. | |
| **SIMILITUDES CON LA INVESTIGACIÓN** | |
| La principal semejanza con esta investigación es que utiliza la metodología Scrum para llevar a cabo un proyecto de equipos, es decir, hace uso de los lineamientos que plantea esta metodología para evaluar las ventajas y desventajas en el uso. | |
| **DIFERENCIAS CON LA INVESTIGACIÓN** | |
| La diferencia fundamental con esta investigación, se centra en que a partir de una caracterización de los elementos metodológicos del proceso de construcción de software en el centro de informática de la Universidad de Nariño, se realizará una contrastación con los de Scrum, para posteriormente formular una propuesta de trabajo propia para dicha dependencia. | |

Fuente: La presente investigación - 2017

Tabla 3. Aplicabilidad de Competisoft a partir de un método ágil como Scrum. Un caso práctico

| **Título antecedente** | **Aplicabilidad de Competisoft a partir de un método ágil como Scrum. Un caso práctico. (Martinez, Ramon, & Bertone, 2012)** |
|---|---|
| **OBJETIVO GENERAL** | |
| La implementación del proyecto Competisoft – Mejora de procesos para fomentar la competitividad de la pequeña y mediana industria del software de Iberoamérica | |

| |
|---|
| en una organización especifica enfocada a arquitectura, desarrollo e implementación de sistemas informáticos, y que utilice una metodología ágil (Scrum) para facilitar mejoras en cuanto a agilidad con el fin de evolucionar y alcanzar un proceso robusto que conlleve a la calidad deseada. |
| **RESULTADOS** |
| Formalizar un proceso metodológico en el desarrollo de software, lo cual genera la aplicación de un método coordinado, forma y ordenado en el hacer de las funciones, teniendo en cuenta una clara definición del papel o rol dentro del proceso y sus responsabilidades. Las reuniones de retrospectiva permitieron formar un equipo de trabajo colaborativo y asertivo, permitiendo la mejora continua de la calidad de los procesos. |
| **CONCLUSIONES** |
| Uno de los objetivos de las empresas de software es ampliar su mercado y una de las estrategias es tener una relación muy estrecha con el cliente, satisfaciendo sus requerimientos. En la actualidad se hace más visible la necesidad de certificaciones internacionales por parte de pequeñas y medianas empresas de desarrollo de software, y para ello se debe tener implementado una metodología que asegure la calidad de los productos, por tal razón en esta investigación se demostró que el uso de Competisoft y Scrum se enfocan a conseguir mejoras continuas y a su sencilla aplicación. |
| **SIMILITUDES CON LA INVESTIGACIÓN** |
| La principal semejanza con esta investigación es que utiliza la metodología Scrum para llevar a cabo un proyecto de desarrollo de software, es decir, hace uso de los lineamientos que plantea esta metodología para evaluar las ventajas y desventajas en el uso. |
| **DIFERENCIAS CON LA INVESTIGACIÓN** |
| La diferencia fundamental con esta investigación, se centra en que a partir de una caracterización de los elementos metodológicos del proceso de construcción de software en el centro de informática de la Universidad de Nariño, se realizará una contrastación con los de Scrum, para posteriormente formular una propuesta de trabajo propia para dicha dependencia. |

Fuente: La presente investigación - 2017

Tabla 4. *Scrum practice mitigation of global software development coordination challenges: A distinctive advantaje?*

| **Título antecedente** | ***Scrum practice mitigation of global software development coordination challenges: A distinctive advantaje? (Bannerman, Hossain, & Jeffery, 2012)*** |
|---|---|
| **OBJETIVO GENERAL** | |
| Definir qué ventajas se ven reflejadas en la mitigación de problemas de coordinación de proyectos de desarrollo de software, al ser implementada la metodología Scrum dentro de 4 proyectos a nivel mundial. | |

| |
|---|
| **RESULTADOS** |
| Scrum se adaptó para reducir al máximo la necesidad de reuniones con todos los miembros al tener el inconveniente de cambio de horario por ubicación geográfica, y las reuniones que fueron necesarias se desarrollan dentro de un ámbito multicultural, para mitigar diferencias entre normas, valores y perspectivas nacionales. Además las reuniones se hicieron con ayuda de TIC´s. Se puede decir que Scrum tiene una ventaja grande al mitigar efectos de distancia socio-culturales dentro del desarrollo de un proyecto. |
| **CONCLUSIONES** |
| Con el método Scrum los proyectos distribuidos pueden verse muy beneficiados en cuanto a la mitigación de problemas de coordinación generados por la distancia geográfica y sociocultural, es decir para los administradores de proyectos en los casos estudiados, esto era algo que los métodos tradicionales no les daban solución. |
| **SIMILITUDES CON LA INVESTIGACIÓN** |
| La principal semejanza con esta investigación es que utiliza la metodología Scrum para llevar a cabo el desarrollo de dichos proyectos, haciendo uso de los lineamientos que plantea esta metodología para evaluar las posibles ventajas y desventajas en su aplicación. |
| **DIFERENCIAS CON LA INVESTIGACIÓN** |
| La diferencia fundamental con esta investigación, se centra en que a partir de una caracterización de los elementos metodológicos del proceso de construcción de software en el centro de informática de la Universidad de Nariño, se realizará una contrastación con los de Scrum, para posteriormente formular una propuesta de trabajo propia para dicha dependencia. |

Fuente: La presente investigación - 2017

Tabla 5. *Introducing Scrum in companies in Norway: A case study*

| | |
|---|---|
| **Título antecedente** | ***Introducing Scrum in companies in Norway: A case study.*** |
| **OBJETIVO GENERAL** | |
| Describir cómo se desarrolla la introducción de Scrum en tres empresas noruegas de desarrollo de software, la investigación se basa en entrevistas que fueron analizadas con el fin de identificar las problemáticas de la implementación de Scrum, el manejo de equipos y roles. | |
| **RESULTADOS** | |
| Al introducir Scrum de forma repentina se puede generar un proceso doloroso, por lo tanto es necesario analizar efectos secundarios y así adaptar el método a su propio contexto. Dentro de la investigación se vio que a pesar de la motivación de gerencia el cambio de cultura es un poco complicado, de todas maneras los trabajadores empezaron a tener días de trabajo más estructurados, con mayor control en sus labores y menos horas extras. | |

| |
|---|
| **CONCLUSIONES** |
| Es muy importante que antes de introducir cualquier metodología, en este caso Scrum, se deba realizar un estudio previo, de cómo y cuándo generar el cambio, desde ser paulatino, y adecuado para cada organización y cultura, ya que el cambio drástico puede afectar la productividad. |
| **SIMILITUDES CON LA INVESTIGACIÓN** |
| La principal semejanza con esta investigación es que utiliza la metodología Scrum para ser involucrada dentro de una empresa y aplicada en sus proyectos, es decir, hace uso de los lineamientos que plantea esta metodología para evaluar las ventajas y desventajas en el uso y el impacto dependiendo del tipo de implementación. |
| **DIFERENCIAS CON LA INVESTIGACIÓN** |
| La diferencia fundamental con esta investigación, se centra en que a partir de una caracterización de los elementos metodológicos del proceso de construcción de software en el centro de informática de la Universidad de Nariño, se realizará una contrastación con los de Scrum, para posteriormente formular una propuesta de trabajo propia para dicha dependencia. |

Fuente: La presente investigación - 2017

Tabla 6. Implementación de un sistema para el control de activos ISOPTEC, bajo el estándar ITIL y metodología ágil Scrum

| | |
|---|---|
| **Título antecedente** | **Implementación de un sistema para el control de activos ISOPTEC, bajo el estándar ITIL y metodología ágil Scrum. (May, Morales, Marrufo, & Martín, 2013)** |
| **OBJETIVO GENERAL** | |
| Mostrar el desarrollo de una aplicación para el manejo y control de activos fijos de las Universidades Tecnológicas aplicando la metodología de desarrollo Scrum así mismo dejando el sistema abierto para en un futuro implementar más funcionalidades y hacer de este un sistema más completo. | |
| **RESULTADOS** | |
| Al aplicar la metodología Scrum se tuvo como resultados que la curva de aprendizaje con un equipo de desarrollo nuevo fue relativamente rápida y así mismo al proceso de documentación el cual es de gran ayuda para futuros proyectos, además como uno de los resultados más importantes es que la implementación del sistema solo tuvo un desfase del 3% lo cual hace del proyecto y de la metodología usada algo que genera calidad en el producto hacia el usuario. | |

| **CONCLUSIONES** |
|---|
| Al realizar el desarrollo con la metodología Scrum se llegó a cumplir con el 100% de los requerimientos por parte del cliente entregando un programa funcional, en donde solo fue necesaria la aplicación de 6 Sprints logrando cumplir con todo lo necesario para el proyecto. |
| **SIMILITUDES CON LA INVESTIGACIÓN** |
| La principal semejanza con esta investigación es que utiliza la metodología Scrum para elaborar un producto software, es decir, hace uso de los lineamientos que plantea esta metodología para evaluar las ventajas y desventajas en el uso. |
| **DIFERENCIAS CON LA INVESTIGACIÓN** |
| La diferencia fundamental con esta investigación, se centra en que a partir de una caracterización de los elementos metodológicos del proceso de construcción de software en el centro de informática de la Universidad de Nariño, se realizará una contrastación con los de Scrum, para posteriormente formular una propuesta de trabajo propia para dicha dependencia. |

Fuente: La presente investigación - 2017

Tabla 7. Metodología adaptativa basada en Scrum: Caso empresas de la Industria de Software en San Juan de Pasto – Colombia

| **Título antecedente** | **Metodología adaptativa basada en Scrum: Caso empresas de la Industria de Software en San Juan de Pasto – Colombia. (Hernández, Martínez, Argote, & Coral, 2015)** |
|---|---|
| **OBJETIVO GENERAL** | |
| Consolidar una metodología adaptativa para la construcción de software que incluya Scrum en las empresas de la Industria de Software en Pasto. | |
| **RESULTADOS** | |
| Los resultados obtenidos en la caracterización del proceso de construcción de software, permitió identificar que las fases que se lleva a cabo son generales y muy básicas e imprescindibles y no se observa que existan actividades claramente definidas. Lo cual se refleja en las dificultades principalmente para formular objetivos del proyecto, establecer roles y definir los entregables. De ahí se establecieron similitudes y diferencias y se definieron los elementos de intervención que se tendrán en cuenta para elaborar la propuesta en la cual se identificaría de qué manera Scrum puede aportar a cada elemento metodológico definido en la investigación (Etapa, actividad, rol, artefacto, herramienta y lineamiento). | |

| CONCLUSIONES |
|---|
| La investigación se trabajó con una muestra de 10 empresas de 21 de la industria de software de Pasto – Nariño, las cuales decidieron colaborar con la investigación. Un alto número de ellas definen sus procesos de construcción de software dentro de fases generales, sin entrar en especificación de las actividades, lo cual trae como consecuencia dificultades en asignación de roles, formulación de objetivos y definición de los entregables.<br><br>La propuesta adaptativa basada en Scrum, se elabora a partir del análisis de similitudes y diferencias por cada elemento metodológico entre las empresas de la Industria de Software en Pasto y los principios definidos en Scrum. |
| **SIMILITUDES CON LA INVESTIGACIÓN** |
| La principal semejanza con esta investigación es consolidar una metodología adaptativa para la construcción de software basada en Scrum. |
| **DIFERENCIAS CON LA INVESTIGACIÓN** |
| La diferencia fundamental con esta investigación, se centra en que a partir de una caracterización de los elementos metodológicos del proceso de construcción de software en el centro de informática de la Universidad de Nariño, se realizará una contrastación con los de Scrum, para posteriormente formular una propuesta de trabajo propia para dicha dependencia e implementarla en el desarrollo de un producto software. Además, la gestión del proyecto a través de la metodología propuesta, será soportada por herramientas de software libre. |

Fuente: La presente investigación - 2017

Tabla 8. KOWLAN: Una experiencia Scrum en un entorno de innovación

| **Título antecedente** | **KOWLAN: Una experiencia Scrum en un entorno de innovación (García-Algarra & González-Ordás, 2010)** |
|---|---|
| **OBJETIVO GENERAL** | |
| Aplicar una metodología ágil en el desarrollo de un proyecto llamado KOWLAN que consiste en dar diagnostico automático a fallas dentro de la red de datos MACROLAN (Telefónica España), con el fin de romper el enfoque tradicional | |
| **RESULTADOS** | |
| Las reuniones periódicas dio como resultado detección temprana de problemas, y compartir conocimientos de forma más ágil, además que incremento la colaboración dentro del equipo. | |
| **CONCLUSIONES** | |

| |
|---|
| Scrum es muy útil en ambientes con altos niveles de incertidumbre, incrementa el trabajo en equipo de manera considerable, facilitando aprendizaje y rendimiento, por tal razón Scrum puede ser aplicado en diferentes tipos de proyectos, ya que posee gran flexibilidad en cuanto a su aplicación. |
| **SIMILITUDES CON LA INVESTIGACIÓN** |
| La principal semejanza con esta investigación es que utiliza la metodología Scrum para llevar a cabo un proyecto de desarrollo de software, es decir, hace uso de los lineamientos que plantea esta metodología para realizar el aporte al eje metodológico. |
| **DIFERENCIAS CON LA INVESTIGACIÓN** |
| La diferencia fundamental con esta investigación, se centra en que a partir de una caracterización de los elementos metodológicos del proceso de construcción de software en el centro de informática de la Universidad de Nariño, se realizará una contrastación con los de Scrum, para posteriormente formular una propuesta de trabajo propia para dicha dependencia e implementarla en el desarrollo de un producto software. Además, la gestión del proyecto a través de la metodología propuesta, será soportada por herramientas de software libre. |

Fuente: La presente investigación - 2017

Tabla 9. *Agile principles and achievement of success in software*

| | |
|---|---|
| **Título antecedente** | ***Agile principles and achievement of success in software development: A quantitative study in Brazilian organizations*** **(de Souza Bermejo, y otros, 2014)** |
| **OBJETIVO GENERAL** | |
| Identificar y analizar los perfiles de las organizaciones Brasileñas que producen software utilizando principios ágiles y examinar sus casos de éxito. | |
| **RESULTADOS** | |
| Se realizó una encuesta la cual contenía 16 preguntas, las cuales estaban divididas en 2 fases un perfil demográfico y un análisis del uso de los principios de desarrollo de software por la organización. En la primera parte los encuestados informaron de sus áreas de práctica profesional, sus puestos de trabajo dentro de la organización, el número de empleados en sus empresas, clasificación administrativa de su organización, los tipos de productos desarrollados en sus organizaciones y si se utilizan metodologías ágiles. La segunda parte del análisis comprendía el uso de los principios de desarrollo de software por parte de la organización, indicando en una escala de 6 ítems su punto de vista sobre la utilización de principios agiles en sus organizaciones. Todas las regiones del país fueron representadas en la muestra. En cuanto al | |

tiempo de uso de metodologías ágiles, las respuestas mostraron que el 26,7% de las organizaciones adopta metodologías agiles desde hace dos años, el 22,2% desde hace un año, el 15,4% desde hace tres años, el 11,7% desde hace 5 años y el 11,0% desde hace 4 años. En cuanto a la etapa de madurez en relación con el uso de metodologías ágiles, de acuerdo con las percepciones de los encuestados, el 22,2 de organizaciones está en la etapa de madurez de la utilización de prácticas ágiles en los procesos repetibles, el 21,8% en la etapa de madurez de la utilización de prácticas ágiles ad hoc, el 21,0% se encuentran en la etapa de madurez de la utilización de prácticas ágiles definidas, el 20,0% en la etapa de madurez del uso administrado y el 14,9% en la etapa de madurez del uso optimizado de las prácticas ágiles.

**CONCLUSIONES**

El objetivo de este estudio fue identificar y analizar los perfiles de las organizaciones brasileñas que producen software, utilizando las experiencias empíricas de los encuestados con respecto al uso de los principios ágiles y los casos de éxito en el desarrollo de software. En resumen, este estudio contribuye a una mejor comprensión de la utilización de los principios ágiles en el contexto del desarrollo, asociado con la obtención de éxito en la producción de software. La investigación en esta área es de gran importancia para el desarrollo de un mejor comprensión, que conduzca a las organizaciones a aumentar las inversiones en recursos, esfuerzos y principios agiles para alcanzar la excelencia en los procesos y productos de software.

**SIMILITUDES CON LA INVESTIGACIÓN**

La principal semejanza con esta investigación es que hace un análisis de los perfiles de organizaciones, tomando como base los principios agiles, que a su vez también hacen parte del marco de trabajo de la metodología Scrum a utilizar en el desarrollo de este proyecto.

**DIFERENCIAS CON LA INVESTIGACIÓN**

La diferencia fundamental con esta investigación, se centra en que a partir de una caracterización de los elementos metodológicos del proceso de construcción de software en el centro de informática de la Universidad de Nariño, se realizará una contrastación con los de Scrum, para posteriormente formular una propuesta de trabajo propia para dicha dependencia e implementarla en el desarrollo de un producto software. Además, la gestión del proyecto a través de la metodología propuesta, será soportada por herramientas de software libre.

Fuente: La presente investigación - 2017

## 1.2 TITULO

Metodología adaptativa basada en Scrum, apoyada por herramientas de software libre para el Centro de Informática de la Universidad de Nariño.

## 1.3 PROBLEMA DE INVESTIGACIÓN

**1.3.1 Descripción del problema.** La construcción de software es un proceso conformado por pasos ordenados para solucionar un problema elaborando un producto software como parte de esta. Este proceso se categoriza como complejo por la gran cantidad de factores que inciden en su desarrollo. Así mismo, el método que se utilice para la gestión de la construcción de software, se basa en metodologías, ágiles y tradicionales; que permiten estructurar un marco de trabajo para el ciclo de vida del proyecto. Pero muchas veces dentro de las organizaciones por su dinámica o sus particularidades, estos procesos, se ven entorpecidos por factores independientes al uso o no de una metodología, situación que se hace evidente en el Centro de Informática de la Universidad de Nariño - CI. Actualmente, el CI cuenta con un grupo de 6 a 7 ingenieros, los cuales tiene a su cargo el desarrollo, mantenimiento y soporte de todas las aplicaciones que funcionan y nuevos requerimientos solicitados por las diferentes unidades académico-administrativas de la Universidad, en donde cada funcionario es responsable de un proyecto en particular (Castillo, 2016). Cuando un Ingeniero, renuncia al cargo, según Castillo Eraso (2016), en ocasiones la vacante no se cubre, y las funciones y responsabilidades son transferidas al personal existente, generando sobrecarga laboral, con consecuencias como estancamientos y retrasos en la producción de software. Este aspecto, está generando una alta rotación del personal del CI. Para la vinculación de nuevo personal, de acuerdo con Castillo Eraso (2016), este tiene que llegar a aprender en el ejercicio de sus funciones; es decir, no existen unos procesos y procedimientos claramente definidos como lineamientos por parte del CI, sobre el quehacer de los ingenieros para la gestión en la construcción de software. Esta novedad, acarrea como consecuencia, una prolongación en la curva de aprendizaje del funcionario para iniciar a ser productivo en el CI. Además, la falta de recopilación de datos para establecer indicadores de desempeño en el desarrollo del software, hace que las decisiones no sean informadas, sino basadas en la experiencia y en la percepción del director del CI (Castillo, 2016).

Por otra parte, según Castillo Eraso (2016), el aseguramiento de la calidad de los productos software que se construyen en el CI, se simplifica a la prueba-error, donde el mismo programador y el cliente son quienes detectan los defectos. Además, no

se lleva una bitácora de registro de los defectos detectados y el tratamiento que se les ha dado. Estos errores, se corrigen en la marcha, evidenciando la ausencia de versionamiento en los diferentes productos desarrollados. En cuanto a la documentación de los productos software, Castillo Eraso (2016), afirma que hay un bajo porcentaje de manuales de usuario y programador, que distan de la realidad, por lo cambios continuos que se hace al software y la falta de actualización de los manuales. La práctica que se está utilizando para hacer mantenimiento al software consiste en analizar el código y determinar cuál es su funcionalidad, para comprender el dominio y proseguir con el mantenimiento o desarrollo. Además, según palabras de Castillo Eraso (2016) existen sistemas antiguos que fueron desarrollados de forma aislada e independiente por funcionarios que ya no están, que hasta la fecha no han podido ser actualizados e integrados al sistema de información de la Universidad, lo que genera desgaste administrativo y redundancia en la ejecución de procesos y de la información que estos gestionan, lo que se refleja en retrasos al momento de presentar informes a los entes de control y en la generación información para la toma de decisiones oportuna y eficiente en pro del cumplimiento de las metas de la Universidad.

Los síntomas anteriormente descritos, se presentan porque el trabajo de los ingenieros en el CI, es de manera individual, no existe una forma de trabajo claramente establecida, el desempeño obedece a la experiencia y conocimiento de quién esté realizando el soporte, mantenimiento o nuevo desarrollo. Estas causas, se derivan en una alta carga laboral para los ingenieros, debido a que en el CI, todas los requisitos de usuario por desarrollar son urgentes (Castillo, 2016). Así mismo, se logra identificar que no existe compromiso y apoyo, por parte de la alta dirección de la Universidad de Nariño sobre la importancia del CI como una dependencia que sirve de soporte a los proceso de negocio, tanto académicos como administrativos (Castillo, 2016).

Lo anterior permite diagnosticar que el CI de la Universidad de Nariño está desprovisto de una forma de trabajo estratégica y claramente definida, que le permita gestionar el proceso de soporte, mantenimiento y construcción de software, para dar respuesta a las unidades académico-administrativas en los requerimientos cambiantes y urgentes.

De continuar con la problemática descrita en los párrafos anteriores, la Universidad, a través de las unidades académico-administrativas, se verá relegada en el uso de la tecnología como un aliado estratégico para ayudar al cumplimiento de sus funciones sustantivas, el CI tenderá a perder importancia como la unidad que articula e integra los procesos de negocio de la universidad con la infraestructura tecnológica. Al continuar trabajando de la manera como se viene realizando las

labores de soporte, mantenimiento y construcción de software, por los avances tecnológicos y por las necesidades crecientes de software en el mundo actual, se necesitará más personal, lo que se verá reflejado en el incremento de los costos de operación del CI. De igual manera, las unidades académico-administrativas de la Universidad, tendrán mayores requisitos de información, que al no ser satisfechos por el CI, se verán expuestas a la toma de decisiones sin contar con información o a subcontratar personal que apoye estas labores, traduciéndose en un incremento de sus costos de operación.

**1.3.2 Formulación del problema.** ¿Cómo consolidar una metodología para el soporte, mantenimiento y construcción de software, basada en Scrum y apoyada por herramientas de software libre para el Centro de Informática de la Universidad de Nariño?

## 1.4 OBJETIVOS

**1.4.1 Objetivo General.** Consolidar una metodología para el soporte, mantenimiento y construcción de software, basada en Scrum y apoyada por herramientas de software libre para el Centro de Informática de la Universidad de Nariño.

### 1.4.2 Objetivos Específicos

- Caracterizar los elementos metodológicos, del proceso de soporte, mantenimiento y construcción de software del Centro de Informática de la Universidad de Nariño y los definidos en el marco de trabajo Scrum.

- Describir de manera comparativa los elementos metodológicos del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los de Scrum.

- Formular una propuesta de trabajo basada en Scrum y apoyada por herramientas de software libre para el proceso de construcción de software del Centro de Informática de la Universidad de Nariño.

- Aplicar la propuesta de trabajo basada en Scrum y apoyada por herramientas de software libre para el desarrollo de un producto software en el Centro de Informática de la Universidad de Nariño.

## 1.5 JUSTIFICACIÓN

Este trabajo investigativo es interesante porque, al caracterizar los elementos metodológicos del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los definidos en el marco de trabajo Scrum se conocerá y analizará la realidad actual sobre la forma de trabajo del CI, lo que permitirá tener un panorama claro y amplio sobre la dimensión del proceso de soporte, mantenimiento y desarrollo de software; y de los recursos técnicos y humanos involucrados. A demás no solo se plantearan elementos teóricos sobre una propuesta de trabajo basada en Scrum, sino que se evaluará su aplicación en la construcción de un producto software.

Esta investigación es útil, ya que al realizar una comparación entre los elementos metodológicos del proceso de construcción de software del CI y los de Scrum, se logrará identificar cuáles son las actividades, elementos, procesos o pasos que existen en común, definiendo cómo se puede involucrar a Scrum gestionado por herramientas de software libre en el CI, de manera dinámica, eficiente y con un impacto mínimo en su aplicación, cuyo proceso será descrito en una propuesta basada en este marco de trabajo, que será una herramienta importante y estratégica para el proceso de negocio de esta dependencia, en donde el contar con una metodología ágil ajustada a las características y contexto del CI, hará que su forma de trabajo sea eficiente y estratégica, lo que posibilitará el cumplimiento en tiempos de desarrollo y calidad del software que soporta los requerimientos cambiantes y urgentes que están inmersos en los procesos de las diferentes unidades académico-administrativas. Además, se contará con indicadores para la toma de decisiones por parte del director, las cuales ya no se basarán únicamente en su experiencia y percepción. También el hecho de probar la propuesta en el desarrollo de un producto software, generará información valiosa que será útil para determinar si contribuye al mejoramiento de la forma de trabajo y la eficiencia en el soporte, mantenimiento y construcción del software y demás tareas que debe ejecutar a diario esta dependencia como eje integrador de los procesos de negocio de la Universidad con la infraestructura tecnológica.

Esta investigación es novedosa, ya que hasta el momento, no se ha realizado ningún estudio, que permita caracterizar los elementos metodológicos de soporte, mantenimiento y construcción de software del CI de la Universidad de Nariño, y la

formulación de una propuesta que los integre con los lineamientos dados por el marco de trabajo Scrum y determine el nivel de aporte de herramientas de software libre a este proceso, dejando una base para futuras investigaciones o estudios, en donde se quiera integrar elementos de otras metodologías en pro de entregar productos con el máximo valor posible, productiva y creativamente.

## 1.6. MARCO TEÓRICO

**1.6.1. Metodología adaptativa basada en Scrum.** Para poder definir lo que es, el sentido y significado de una metodología adaptativa basada en Scrum, se iniciará realizando una descripción conceptual de la manera como este concepto surge y la acepción que se tendrá para el desarrollo de este trabajo investigativo.

La construcción de software es un proceso conformado por pasos ordenados para solucionar un problema u obtener un producto, específicamente un producto software que se utilizará para resolver un problema planteado. Este proceso puede convertirse en algo complejo que depende de sus características y alcance, por tal razón existen tres categorías basándose en el tamaño o costo: pequeño, mediano y gran porte, así mismo para estimar cada proyecto existen diferentes tipos de metodologías, ágiles y tradicionales.

La construcción de software presenta un conjunto de problemas que hacen que los proyectos fracasen, o finalicen con retos y desafíos, que no cumpla objetivos o genere fallos inaceptables. Se estima que, del total de proyectos software grandes emprendidos, un 28% fracasan, un 46% caen en severas modificaciones que lo retrasan y un 26% son totalmente exitosos (PRESSMAN, 2003), y muy poco frecuente es por fallas técnicas, generalmente la razón es por la falta de aplicación de una metodología efectiva o proceso de desarrollo, lo que ha llevado en las últimas décadas a mejorar dichas metodologías e incluso a crear nuevas concientizando con ello a los profesionales que se encargan de aplicarlas, ya que es muy normal que la metodología para la construcción de software sea resultado de un hibrido que reúne etapas y pasos de procesos anteriores y de criterios propios.

Teniendo en cuenta el artículo de ECURED (2016), el proceso de desarrollo puede llegar a involucrar variadas y numerosas tareas, iniciando lógicamente desde la parte administrativa, técnica, la gestión y el gerenciamiento, pero es casi rigurosamente estricto seguir o cumplir ciertas etapas mínimas, como la elicitación,

especificación y análisis de requisitos**,** Diseño**,** Codificación**,** Pruebas**,** Instalación o despliegue y Mantenimiento.

En este orden de ideas, se hace necesario utilizar una metodología que además de incluir las etapas y actividades, del proceso de software; tenga en cuenta los aspectos relacionados con la administración del proceso. Metodologías para la construcción de software han existido desde hace varios años, y se han venido incorporando nuevas desde enfoques que buscan dar respuesta al término que se ha acuñado como “crisis del software” (NAUR & RANDELL, 1968).

Un enfoque que ha sido la base para el desarrollo de nuevas metodologías para la construcción de software es el agilísmo planteado por Beck y otros (2001). Para Canós, Letelier y Penadés (2003), la agilidad en un proceso, se presenta cuando el desarrollo del software es incremental, cooperativo (continua comunicación), sencillo y adaptable. Estas cualidades, se presentan dentro de periodos cortos de tiempo, en donde se aplica planeación, análisis, diseño, codificación, revisión y documentación, generando un incremento del producto sin errores; a estos periodos se les llama iteración. Para Canós, Letelier y Penadés (2003) estas metodologías son propias cuando un proyecto tiene requisitos poco definidos o demasiado cambiantes, existen normas de código que todo equipo de desarrollo debería cumplir, pero al tener sobrecarga de trabajo es muy complicado que se consiga esa regulación, se quiere que la calidad se refleje en la satisfacción del cliente frente al producto, no se desea generar agotamiento en el equipo de trabajo; y se quiere soportar el proceso de software con herramientas, es decir, una gran inversión en planificación y herramientas de gestión.

Una de las metodologías ágiles para la gestión de proyectos que ha sido ampliamente utilizada en los últimos años es Scrum**.** Schwaber y Sutherland (2013, pág. 4) manifiestan que Scrum es un tipo de metodología ágil orientada a la gestión de desarrollo de software, buscando maximizar la ganancia a la inversión de cada empresa que aplica este proceso.

Se fundamenta en enfocarse en los productos funcionales más importantes para el cliente y verificación continua, nivel de adaptación e innovación; es un proceso en donde se utilizan y aplican mejores prácticas para el trabajo colaborativo, en equipo y así maximizar el buen resultado, consiguiéndolo al apoyar las prácticas unas en otras (PROYECTOSAGILES.ORG). Una de las principales características es la entrega parcial y regular del producto final, según el beneficio hacia el cliente, por tal razón es indicado para entornos inestables, en donde se requieren prontos resultados, con requisitos cambiantes y/o pocos definidos, priorizando en la competitividad, productividad y flexibilidad.

Uno de los grandes beneficios se desprende cuando el cliente mira que su producto crece y se intensifican sus funcionalidades, generando entusiasmo, interés y compromiso frente al proyecto, permitiendo que su interacción sea más continua y pueda modificar o replantear los objetivos o las funciones que se necesitan al inicio de cada periodo o iteración. Se puede definir como otros beneficios, el incremento de la calidad de un producto, las entregas oportunas, el incremento de la productividad del equipo de trabajo, reducción de riesgos y consecuentemente menos costo en cambios, incremento en respuesta y adaptación a cambios, planeación más precisa de tiempo, costo y esfuerzo (SCRUM MANAGER, 2013), lo que es de vital importancia para las empresas de la Industria de Software y para la presente investigación debido a que se debe tener en cuenta todos estos aspectos teóricos, los que debe cumplir la metodología permitiendo así su aplicación en entornos.

**1.6.1.1 Historia de la metodología Scrum.** Basándose en la investigación de Shwaber y Sutherland **(2013)**, la metodología Scrum fue pensada, analizada y definida por Ikujiro Nonaka e Hirotaka Takeuchi, en la década de los años ochenta. Fue la respuesta a la forma en cómo se estaban desarrollando los productos de software en las diferentes empresas de tecnología, tales como, Canon, Honda, Epson, Brother, Hewlett-Packard, entre otras. Inicialmente hicieron una comparación entre la formación Scrum dentro del rugby al reiniciar el juego luego de una falta, con la forma de trabajo en equipo que se estaba diseñando, por tal razón se tomó y definió el nombre Scrum. Todo este análisis fue publicado como artículo en *Harvard Business Review*, en el año 1986 con el nombre "El nuevo juego para el desarrollo de productos", describiendo la experiencia de las empresas nombradas anteriormente al implementar escalabilidad en el desarrollo de productos, y manejo de equipos integrales y auto-organizados.

Como acto seguido, en 1995 el señor Ken Schwaber presentó "*Scrum Development Process*" dentro de OOPSLA 95 *(Object-Oriented Programming Systems & Applications conference*), el cual establece un conjunto de reglas para el desarrollo de software, tomando como base los principios de Scrum, y la experiencia de su autor sobre el desarrollo de Delphi, y Jeff Sutherland en su empresa *Easel Corporation* cuando en 1993 creó el proceso Scrum para aplicar en procesos de desarrollo de software.

**1.6.1.2. Etapas.** Scrum maneja principalmente cuatro fases, las cuales se describe a continuación:

**Pre-Juego.** Se refiere a la planeación de cada iteración, incluye análisis y diseño. En esta fase el cliente debe presentar al equipo de trabajo una lista de requisitos priorizada del proyecto, se resuelven las dudas que puedan surgir y se hace una selección de los requisitos de forma prioritaria con los cuales se debe terminar la iteración que se está iniciando, es decir, se definen cuales requisitos se entregaran en la presente iteración.

Seguido a la definición de cuáles son los requisitos a entregar, el equipo se dedica a elaborar la lista de tareas correspondientes a la iteración, y que son necesarias para el desarrollo de los requisitos con los cuales se comprometieron. En este punto, la estimación se hace de forma conjunta y los propios miembros se auto-asignan sus tareas.

Es importante destacar que de esta fase surge una estimación de coste, cronograma y diseño de las funcionalidades. (SCRUM MANAGER, 2013)

**Juego.** Se deben realizar reuniones diarias de verificación y sincronización, con el fin de que los miembros del equipo puedan inspeccionar el trabajo de los demás miembros, en el caso que exista dependencias de tareas, y revisar el progreso para alcanzar el objetivo e inconvenientes que se están presentando, todo esto es con el fin de realizar adaptaciones que permitan más fácilmente cumplir con el compromiso que se adquirió en la fase anterior. En estas reuniones (max. 15 minutos) se presentan los siguientes interrogantes: ¿Qué se ha hecho desde la anterior reunión de sincronización? ¿Qué haré a partir de esta reunión? ¿Cuáles son los impedimentos que he tenido y cuales posiblemente tendré?

En esta fase principalmente, se presenta el desarrollo de cada sprint, es decir el proceso de ir alcanzando el objetivo de la nueva funcionalidad, teniendo en cuenta que un integrante del equipo denominado *Scrum Master* debe velar para que el equipo logre su compromiso y mantenga productividad y lo hace aminorando obstáculos que el equipo solo no puede resolver y proteger al equipo para que no se vea afectado por interrupciones externas que puedan afectar su compromiso y productividad.

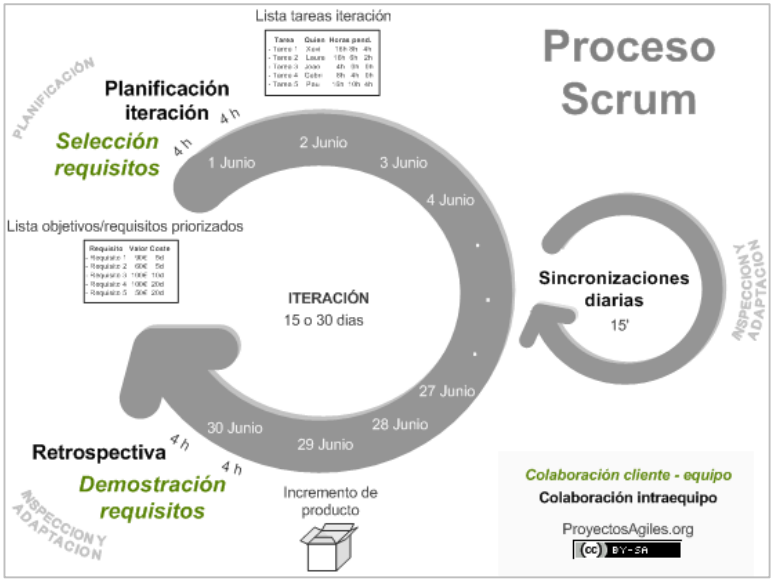
**Post-Juego.** En esta última fase de la iteración se realiza una reunión de revisión consta de dos partes: Demostración, donde se presenta al cliente los requisitos

implementados en la presente iteración (incremental) con el fin de definir resultados y cambios presentados en el proyecto que generan nuevas adaptaciones objetivas que permiten una re-planificación del proyecto. Retrospectiva, espacio donde el equipo debe analizar la forma en la que trabajó la presente iteración y debe también deducir cuales fueron los inconvenientes que no permiten generar un buen curso de trabajo, todo esto con el fin de mejorar continuamente la productividad, además el facilitador debe eliminar los obstáculos que se logran identificar.

Es importante destacar que se debe entregar el incremento con la documentación y pruebas. (SCRUM MANAGER, 2013).

**Cierre.** Al momento de revisar que toda la versión ya está con los objetivos trazados, con los requisitos que se planearon tanto en coste como calidad, y que principalmente todo está alineado para producir una nueva versión, entonces se puede declarar la versión como cerrada.

Figura 1. Proceso Scrum




Fuente: PROYECTOSAGILES.ORG. Cómo gestionar proyectos con Scrum: ¿Qué es Scrum?, s/f, 1p. Disponible en internet, url: <http://www.proyectosagiles.org/que-es-Scrum> (29, 04, 2016)

**1.6.1.3. Eventos.** Según Sutherlan (2013, págs. 9-14), Scrum define los siguientes eventos, los cuales conforman las etapas del proceso:

- ***Sprint*.** El *Sprint* es un grupo de actividades de desarrollo que se lleva a cabo dentro de un periodo establecido (una a cuatro semanas), su duración se define según la complejidad, riesgos y grado de revisión deseada, dentro de este periodo se desarrolla el incremento del producto (útil y entregable), y contiene también el *Sprint Planning Meeting, Daily Scrum, Sprint Review y Sprint Retrospective.* Es importante aclarar, que en cada sprint no se deben realizar cambios que afecten al objetivo del presente *sprint (Sprint Goal)*, el equipo debe ser constante en la duración del periodo *Sprint*, los objetivos de calidad no deben disminuir, el alcance puede ser replantado con el *Product Owner* y el *Development Team* según se va avanzando, y, la duración como se dijo anteriormente, no debe pasar el mes calendario, ya que en este punto el sprint mantiene la adaptación e inspección del progreso al objetivo, limitando riesgos y costos, pero si el sprint abarca más de un mes, el riesgo puede aumentar ya que aumenta su complejidad.

Es claro que el *Product Owner* puede cancelar un sprint, esta decisión la puede tomar con influencia del *Development Team* o del Scrum Master, y la razón principal es cuando el objetivo se queda obsoleto lo cual se puede presentar si cambia la dirección de la organización o el mercado. En el caso de cancelación de un *Sprint*, se debe revisar todos los elementos de la Pila de Producto, y revisar cuales son entregables, y se vuelve a estimar los elementos que no se han desarrollado. Cada sprint contiene un equipo enfocado en:

**Desarrollo**: Definición de los cambios necesarios para la implementación de los requisitos del *backlog*, diseño, desarrollo, implementación, pruebas y documentación de dichos cambios.

**Envoltura**: Generación de la versión presentable con los cambios implementa los requisitos del *backlog*.

**Revisión**: Reunión de los miembros del equipo para realizar la revisión del trabajo y progreso, identificando dudas u obstáculos, con el fin de agregar nuevos elementos al *backlog*.

**Ajuste**: Consolidación de la información de la revisión de las funcionalidades afectadas.

Cada sprint conlleva una revisión en donde el equipo participe de forma completa: La revisión puede incluir a clientes, y otras personas que sean de interés al proyecto. La revisión incluye los sistemas funcionales y los cambios realizados para implementar los elementos del *backlog*. En la revisión se pueden identificar cambios en la forma en la que se implementaron los elementos del *backlog*. La revisión también puede crear o modificar elementos en el *backlog*, cambiando de esta forma el objetivo de las versiones. Se determina la fecha de la siguiente reunión dependiendo del progreso y complejidad.

- ***Sprint Planning Meeting*.** La reunión de Planificación de *Sprint*, se lleva a cabo con el fin de definir cuál será el trabajo a realizar en el siguiente *Sprint*, y lo decide todo el *Scrum Team*. Generalmente para un sprint de un mes, esta reunión tiene una duración de ocho horas, lo cual es proporcional para *sprints* más cortos. Este tipo de reuniones se dividen en:

¿Cuál será el incremento en este *Sprint*?: El *Product Owner* presenta el incremento anterior y la Pila de Producto, de la cual el *Development Team* debe elegir que elementos son capaces para completar en el *Sprint* que se está iniciando, seguido a esto el Equipo Scrum elabora el *Sprint Goal* que es la guía que da la razón del porque el *Development Team* está implementando este incremento.

¿Cómo se logrará ese incremento?: El *Development Team* construye la Pila de *Sprint*, en donde se diseña y estima las actividades que conllevan al desarrollo del incremento, de esta forma puede el equipo darse cuenta si al escoger los elementos de la Pila de Producto el trabajo se vuelve excesivo, por lo tanto se hace un renegociación con el *Product Owner* para crear un compromiso más real en el tiempo. Finalmente se presenta el plan de la Pila de *Sprint* al *Producto Owner* y al *Scrum Master*.

- ***Daily Scrum*.** El Scrum Diario es una reunión llevada a cabo por el *Development Team,* con duración de máximo quince minutos, en donde se mira el estado actual del progreso del incremento, y lo que se hará en las próximas 24 horas. En esta reunión cada miembro del equipo debe explicar que ha logrado desde la última reunión, que hará antes de la próxima reunión y que obstáculos existen.

El papel del *Scrum Master* dentro de este proceso, es el de controlar que se hagan las reuniones, que su duración no sobrepase los 15 minutos, y que solo intervengan los miembros del *Development Team.*

Este tipo de reuniones ayudan a mantener la comunicación, mejorar la necesidad de otras reuniones, identificar y eliminar obstáculos dentro del desarrollo, toma de decisiones rápidas y mejorar el conocimiento del *Development Team* en el proyecto.

- ***Sprint Review*.** La Revisión de *Sprint*, se lleva a cabo al finalizar cada *Sprint*, con el fin de inspeccionar el incremento y actualizar la Pila de Producto si es necesario. El tiempo de esta reunión debe ser de cuatro horas para sprint de un mes, y proporcional para sprints más cortos.

En esta reunión se debe presentar por parte del *Product Owner* que ha sido desarrollado y que no y presenta el estado actual de la Pila de Producto, por parte de los miembros del *Development Team* se presenta un resumen de lo que estuvo bien en el sprint, los obstáculos que se presentaron y como fueron solucionados, además presentan el trabajo que realizaron y responde interrogantes acerca del incremento, finalmente el *Scrum Team* analiza que es lo siguiente por hacer, con el fin de entregar una base para la siguiente reunión de planificación del nuevo sprint.

- ***Sprint Retrospective*.** La reunión de Retrospectiva de *Sprint*, se lleva a cabo luego de la revisión de *Sprint* y antes de la Planificación de *Sprint*, con el fin de construir un plan para crear mejoras que se puedan implementar en el siguiente sprint. La duración de esta reunión debe ser de tres horas para sprint de un mes, y proporcional a sprint más cortos.

En esta reunión se debe revisar cómo se desarrolló el último *sprint* a nivel de personas, relaciones, procesos y herramientas; identificar que elementos fueron bien y que mejoras se pueden aplicar, y finalmente crear un plan para aplicar dichas mejoras.

**1.6.1.4. Actividades.** Cada una de las fases o etapas definidas por Scrum posee ciertos pasos o actividades que permiten alcanzar los objetivos propuestos:

- **Planificación.** Desarrollar de un *backlog* completo. Determinar fecha de entrega y funcionalidades de versiones. Seleccionar la versión para el desarrollo inmediato. Trazar los “paquetes del producto” sobre los elementos del *backlog*. Seleccionar el equipo para el desarrollo de la versión. Evaluación y control de riesgos. Estimar el costo, incluyendo desarrollo, material, marketing, formación y despliegue. Conformidad de la dirección y financiación del proyecto.

- **Diseño y Arquitectura.** Revisar los elementos del *backlog* incluidos en esta versión. Identificar los cambios necesarios para implementar el *backlog*. Acotar la arquitectura del sistema para apoyar el nuevo contexto y necesidades. Identificar problemas del desarrollo o modificaciones. Reunión de revisión de diseño. Cada equipo presenta los cambios para implementar los elementos del *backlog*, e identificar posibles reasignaciones.

- **Pasos del desarrollo (*Sprint*).** Reunión con los equipos para revisar los planes de finalización de versión. Distribución, revisión y ajuste de los estándares de conformidad para el producto. Sprints iterativos hasta que el producto se considera listo para su distribución.

**1.6.1.5. Roles.** Para Schwaber y Sutherland (2013, págs. 5-8), Scrum define los siguientes roles:

- ***Scrum Team*.** El equipo Scrum está conformado por el Dueño del Producto, el Equipo de Desarrollo y el *Scrum Master*. La característica principal de estos equipos es que son auto-organizados y multifuncionales, lo que quiere decir que puedes elegir como hacer su trabajo y además no tienen la necesidad de depender de miembros externos al equipo, todo esto optimiza la creatividad, productividad y flexibilidad.

El equipo es el encargado de entregar de manera incremental mejorando la obtención de una retroalimentación, y asegurando que siempre existirá una versión útil y funcional del producto.

- ***Product Owner*.** El Dueño del Producto se encarga de incrementar el valor del producto y el trabajo del equipo, y una de sus principales funciones es la gestión de la Pila de Producto (*Product Backlog*), de la siguiente manera: tener claridad en los elementos de la Pila; ordenar dichos elementos para lograr los objetivos; asegurar la calidad del trabajo del equipo; asegurar que la pila esté disponible y clara para todo el equipo; asegurar que el equipo entiende a un nivel necesario cada elemento de la Pila.

Si es necesario cambiar la prioridad de un elemento de la pila, el dueño del proyecto es el encargado de analizar el tema y decidir, por lo tanto el equipo debe trabajar estrictamente según la pila que define el *Product Owner*.

- ***Development Team*.** El Equipo de Desarrollo se compone por las personas encargadas de entregar cada incremento del producto, al final de cada sprint. Los miembros de este equipo tienen la independencia de organizar y gestionar su trabajo, incrementando de esta manera la eficiencia y efectividad. Las siguientes son características del equipo de desarrollo:

Auto-organizados: solo el equipo de desarrollo decide cómo convertir los elementos de la Pila a incrementos funcionales.

Multifuncionales: es un equipo con todas las habilidades que se requieren para realizar un incremento.

Desarrolladores: Cada miembro del equipo de desarrollo es igual, son desarrolladores, independiente al trabajo que realice.

Responsable: Pueden existir miembros especializados o con ciertas habilidades especiales en algunas áreas, pero finalmente la responsabilidad es del equipo completo.

Unión: El equipo no puede dividirse en sub equipos encargados de funciones específicas.

El tamaño del equipo de desarrollo no debe ser muy pequeño ni muy grande, debe ser lo suficientemente ágil y capaz de llevar a cabo una carga de trabajo. Cuando un equipo es muy pequeño (tres personas) es muy posible que no se encuentren las habilidades necesarias para completar una iteración correctamente, y cuando son muy grandes (más de nueve) se complica la coordinación.

- ***Scrum Master*.** Es la persona encargada de que la metodología Scrum sea aplicada de forma correcta, ajustándose a la teoría, reglas y prácticas. También se encarga de informar a las personas externas al equipo que interacciones pueden ser útiles o no.

Servicio del *Scrum Master* al *Product Owner:* Definir técnicas para gestionar la Pila más efectivamente; comunicar de forma clara, visión, objetivos y elementos de la Pila al equipo de desarrollo; enseñar al equipo Scrum a crear elementos claros y

concisos de Pila; entender la planificación del producto en entorno empírico; practicar la agilidad; facilitar eventos Scrum según se requieran.

Servicio del *Scrum Master* al *Development Team:* Entrenar al equipo a ser auto-organizado y multifuncional; formar y liderar al Equipo en la creación de productos de alto valor; eliminar impedimentos, u obstáculos al progreso del Equipo; facilitar los eventos de Scrum según se requiera o necesite; entrenar al Equipo en el entorno de organizaciones en donde Scrum no ha sido aplicado completamente.

Servicio del *Scrum Master* a la Organización*:* Liderar y entrenar a la organización en su adopción e implementación de Scrum; planificar implementaciones de Scrum en la organización; ayudar a los interesados dentro de la organización a entender y llevar a cabo Scrum; inducir al incremento de la productividad del Equipo Scrum; Alianzas entre Scrum Masters para incrementar la efectividad de la aplicación de Scrum en una organización.

**1.6.1.6. Artefactos.** Para Schwaber y Sutherlan (Schwaber & Sutherland, 2013, págs. 15-17), Scrum define la pila del producto (product *backlog*), la pila del sprint (sprint *backlog*) y las métricas ágiles.

La Pila de Producto se forma por una lista ordenada de todo lo necesario para el producto, siendo así la fuente de requerimientos para cualquier cambio. El *Product Owner* es el encargado del contenido, disponibilidad y orden de esta Pila, teniendo en cuenta que nunca está completa, ya que está sujeta a cambios, evoluciona de la mano con el producto y el entorno, cambia de forma constante identificando que necesita el producto para llegar a ser adecuado, competitivo y útil.

Esta lista incluye todas las características, funcionalidades, requerimientos, mejoras y correcciones para ser hechos sobre el producto, y se definen con descripción, orden y estimación. El orden de la lista se da por valor, riesgo, prioridad y necesidad, determinando que actividades son de desarrollo inmediato, y para lograr eso se debe tener mayor claridad y detalle en el orden más alto, con estimaciones más precisas.

En cuanto al producto, a medida que su uso genera nuevos cambios llegados desde una retroalimentación con el usuario, la pila crece y nunca deja de cambiar conforme al producto, al mercado y a la tecnología. En el caso que varios equipos Scrum

trabajen para un solo producto, la Pila debe contener un atributo que permita agrupar los elementos.

El dueño del Producto y el Equipo de desarrollo deben realizar el proceso denominado *Grooming*, en donde se agrega detalle, estimación y orden a los elementos de la Pila, de todas maneras el Dueño del Producto puede actualizar los elementos cuando él lo vea necesario y el Equipo de Desarrollo es el encargado de entregar las estimaciones finales dentro de las recomendaciones o instrucciones del Dueño del Producto para conseguir el compromiso fijado.

El seguimiento que debe desarrollar el miembro que ejecuta este rol refiere a la revisión del trabajo total restante en cada final de un sprint y comparando este resultado con el calculado en el sprint anterior, de esa manera se puede visualizar el progreso en el trabajo.

La Pila *Sprint* ésta conformada por los elementos de la Pila de Producto que corresponden para el *Sprint* especificado y por un plan que refleja la entrega de incremento y la consecución del objetivo, el Equipo de Desarrollo define la funcionalidad que será entregada y el trabajo necesario para ello.

El nivel de detalle del plan de la Pila *Sprint* debe ser lo suficientemente claro para que los cambios puedan ser entendidos en el Scrum diario, además el Equipo de Desarrollo puede actualizar la Pila durante la vida del *Sprint* y el aprendizaje del equipo, y si es necesario más trabajo se va agregando a la pila con su estimación correspondiente, y en caso contrario, si un elemento se lo ve innecesario, entonces puede ser eliminado de la Pila *Sprint*.

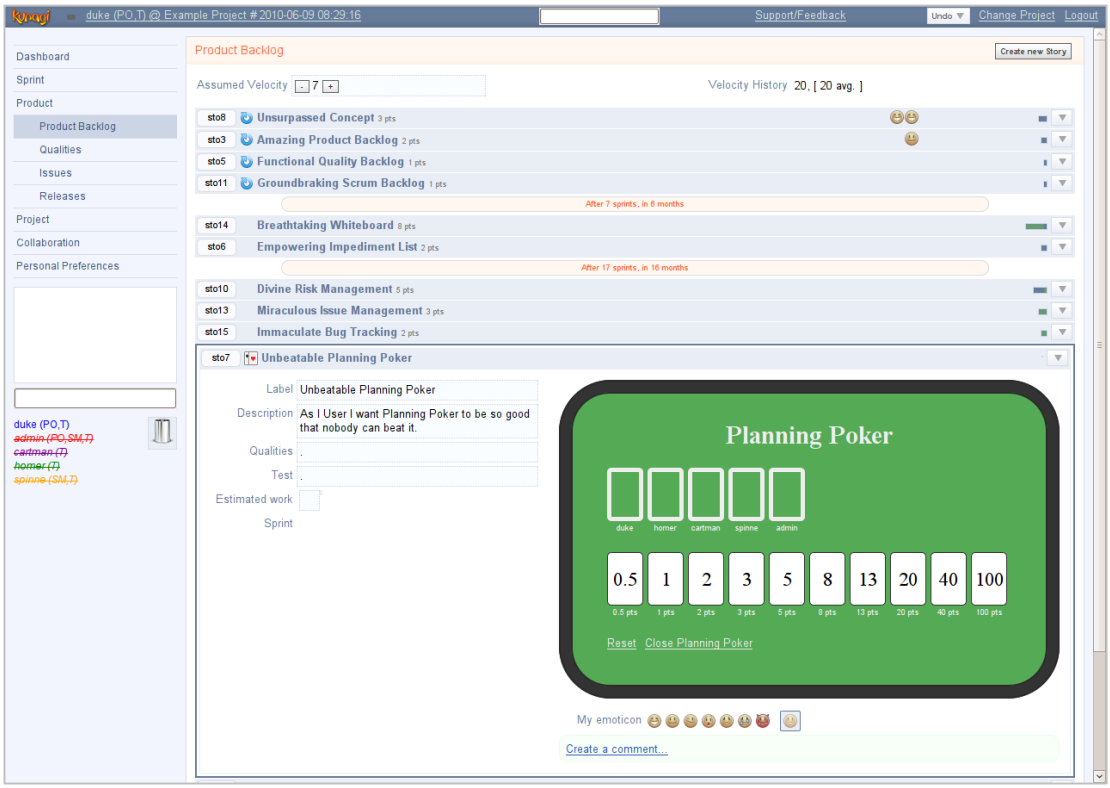
El seguimiento que debe desarrollar el miembro que ejecuta este rol refiere a la revisión del trabajo total restante dentro del sprint, de forma diaria, y comparando este resultado con los calculados a lo largo del sprint, de esa manera se puede visualizar el progreso del equipo de desarrollo dentro del sprint, claro está que no se tiene en cuenta el tiempo utilizado por elemento sino el trabajo restante y sus fechas.

Las métricas ágiles son el elemento que permite recopilar datos y convertirlos en información, como materia prima para la toma de decisiones. Estas permiten definir que medir; y establecer la importancia y la forma como se realizará la medición. Por ejemplo, se puede medir valor de las tareas en desarrollo o el trabajo pendiente por realizar.

**1.6.2. Herramientas para la gestión de proyectos basada en Scrum.** Existen muchas herramientas que ayudan al uso de la metodología Scrum, a continuación se hará una lista de las herramientas libres.

**1.6.2.1. kunagi**. Es una herramienta web que ofrece un sistema de gestión de proyectos basado en Scrum, contiene herramientas colaborativas, un cuadro de mando, un panel interactivo para cada *Sprint* y soporte en estimaciones con *Planning Poker*. No solo ofrece el manejo de los elementos básicos de Scrum, sino también una variedad de datos adicionales.

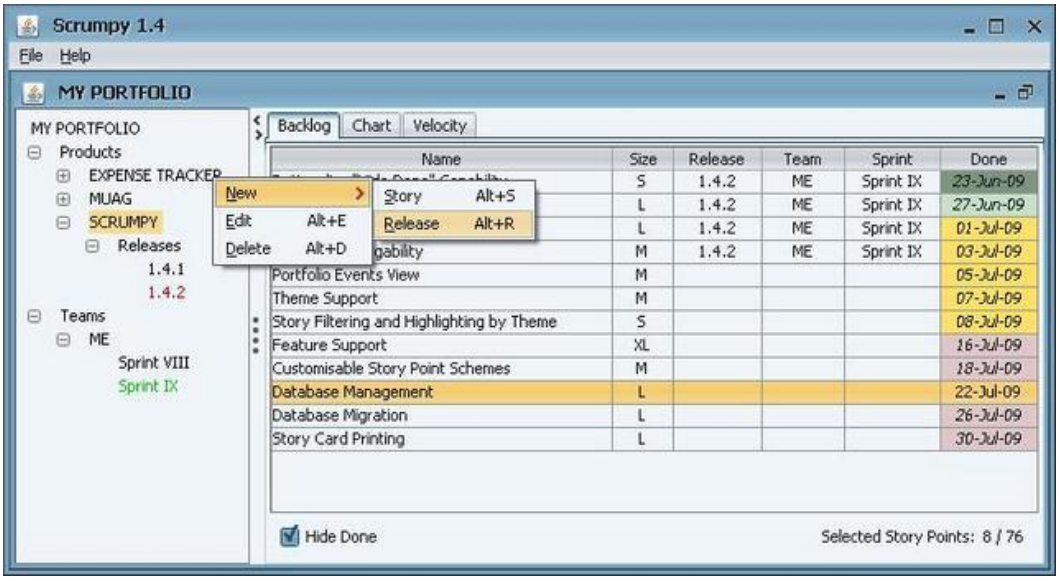
Figura 2. Herramienta Kunagi




Fuente: KUNAGI.ORG. ¿What is Kunagi?, 2010, 1p. Disponible en internet, url: <http://kunagi.org/> (04, 05, 2016)

**1.6.2.2. Scrumpy.** En una aplicación Java diseñada específicamente para ayudar a manejar la Pila de Producto al *Product Owner*, proporcionando una visión significativa a lo largo de las sprints.

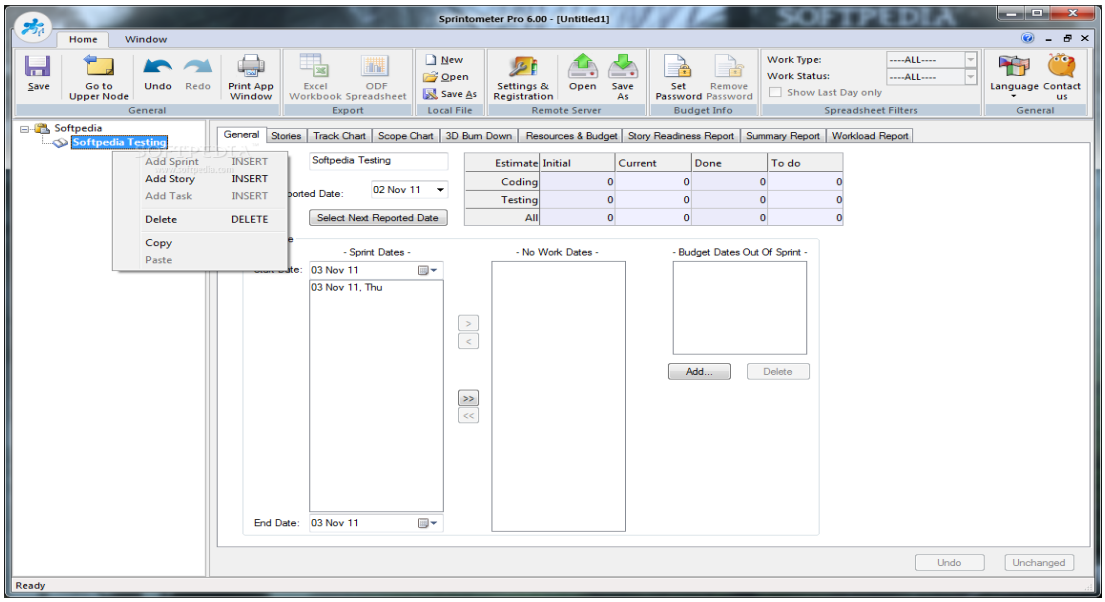
Figura 3. Herramienta Scrumpy




Fuente: USERSTORIES.COM. Scrumpy. s/f, 1p. Disponible en internet, url: <http://www.userstories.com/products/51-Scrumpy> (05, 05, 2016)

**1.6.2.3. Sprintometer.** Es una herramienta con una interfaz gráfica moderna, que permite la administración de proyectos de metodología ágil, Scrum y XP, permite la simplificación de intercambios de datos con programas externos.

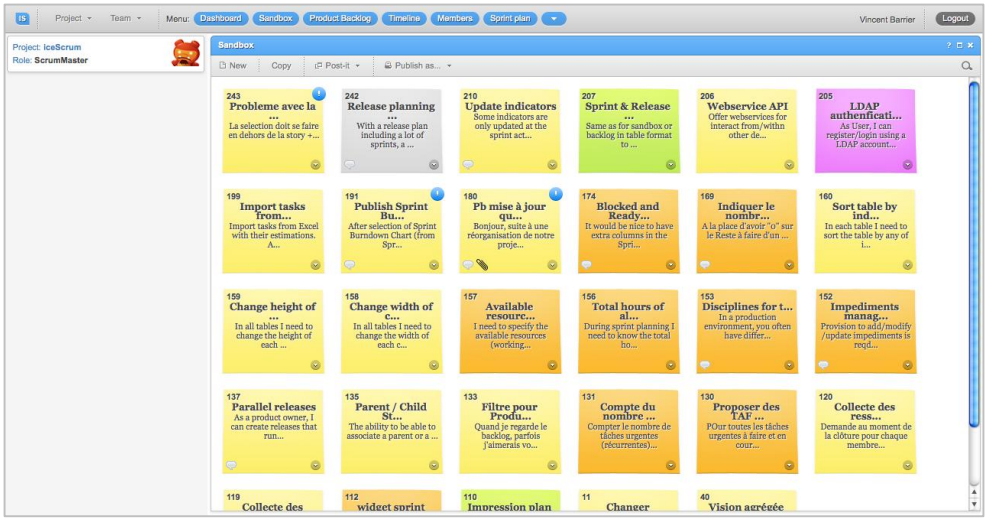
Figura 4. Sprintometer.



Fuente: SPRINTOMETER©. Sprintometer - Scrum & XP project tracking, 2008-2011.Disponible en internet, url: <http://sprintometer.com/> (05, 05, 2016)

**1.6.2.4. IceScrum.** Permite consulta y estimación de historias de usuarios, además agregar elementos a la pila de producto, dividir tiempo en sprints y mover los elementos de la pila de producto a la pila sprint. También soporta a estimaciones con *Planning Poker*.

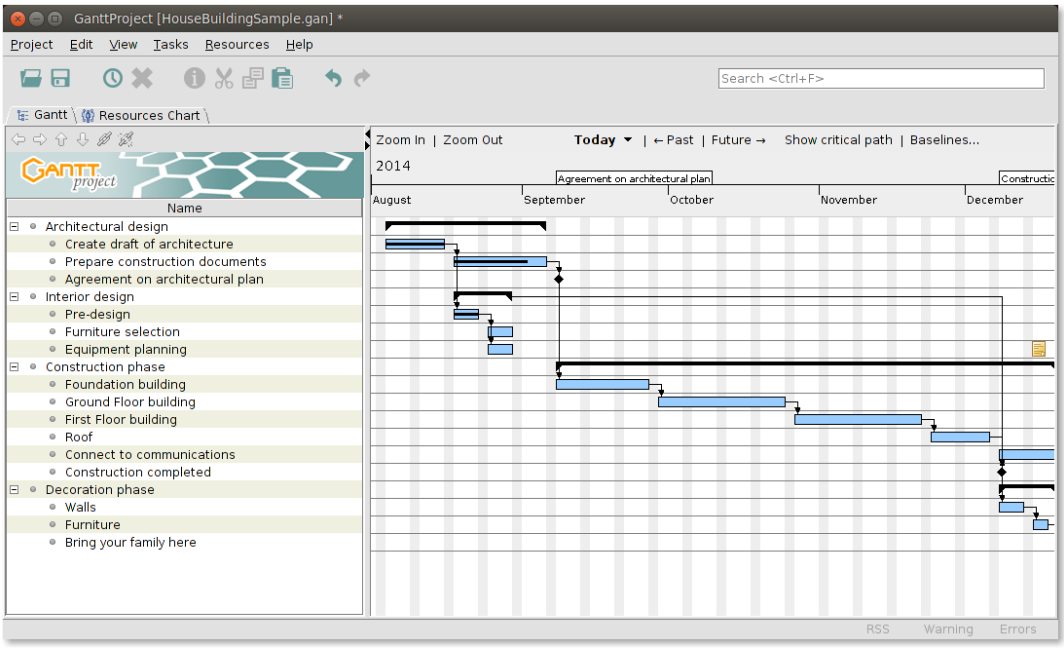
Figura 5. IceScrum




Fuente: ICEScrum, 2016. Disponible en internet, url: <http://www.iceScrum.com/> (06, 05, 2016)

**1.6.2.5. GanttProject.** Es una herramienta multiplataforma para la programación y gestión de proyectos, permite crear diagramas de Gantt posibilitando la visualización de separación de tareas o de las actividades programadas, los eventos o hitos en el desarrollo del proyecto, además de cómo estas se relacionan unas con otras manteniendo la correspondiente jerarquía y dependencia.

Figura 6. Herramienta GanttProject




Fuente: GANTTPROJECT, 2017. Disponible en internet, url: <http://www.ganttproject.biz/img/feature-gantt-chart-big.png> (01, 03, 2017)

## 1.7. METODOLOGÍA

**1.7.1 Paradigma, enfoque y tipo de investigación.** Esta investigación es de corte cuantitativo, porque según Hernández (HERNANDEZ, 2010) representa un proceso secuencial, riguroso y probatorio; parte de una idea que una vez delimitada, se deriva en preguntas de investigación y objetivos. Además, a partir de las preguntas, se establece una hipótesis y se determinan variables.

Posteriormente, se realiza mediciones para que los resultados sean analizados a través del uso de estadística descriptiva y de manera deductiva obtener conclusiones empíricas.

El enfoque para esta investigación es Empírico-analítico, ya que examina el área de estudio y su práctica como un fenómeno que debe ser estudiado de forma objetiva, es decir, a través de una comprensión instrumental y técnica, al estilo de los estudios positivistas (RAMÍREZ RAMÍREZ , pág. 6). Entre los presupuestos que caracterizan este enfoque y se aplican a esta investigación, se destaca que los hechos y

fenómenos que componen la metodología para construir software tienen carácter objetivo, son observables y medibles.

El tipo de investigación es descriptiva, porque según Tamayo (TAMAYO TAMAYO, 2005, pág. 44), describe de manera sistemática las características de una población, situación o área de interés, es decir, especifica las propiedades importantes del evento o situación que se está estudiando. En este caso, se pretende analizar y describir los elementos metodológicos que componen el proceso de construcción de software del Centro de Informática de la Universidad de Nariño. Además, es propositivo ya que, a partir de los resultados obtenidos en la descripción del estudio, se pretende consolidar una metodología para la construcción de software, basada en Scrum y apoyada por herramientas de software libre para el Centro de Informática de la Universidad de Nariño, que fortalezca los procesos de desarrollo de software.

**1.7.2 Línea de investigación.** El presente estudio se enmarca en la línea de investigación correspondiente a Sistemas de Información e Ingeniería de Software.

**1.7.3 Población y muestra.** La población son las Universidades públicas del Sur-Occidente Colombiano. La muestra es no probabilística de tipo intencional, ya que se trabajará en un caso de estudio, que según Gallardo y Moreno (GALLARDO & MORENO, 1999, pág. 108) es conocido como caso típico, debido a que el análisis, de los elementos metodológicos del proceso de construcción de software, se realizará para la Universidad de Nariño, específicamente el Centro de Informática.

**1.7.4 Proceso de investigación.** En siguiente tabla, se categoriza el proceso para cada objetivo específico propuesto en la investigación.

Tabla 10. Proceso de investigación

| Objetivos específicos | Fuente | Técnica de recolección | Instrumento | Técnica de procesamiento | Resultado |
|---|---|---|---|---|---|
| Caracterizar los elementos metodológicos, del proceso de construcción de software del centro de informática de la Universidad de Nariño y los definidos en el marco de trabajo Scrum. | – Centro de Informática de la Universidad de Nariño<br>– Metodología Scrum | – Encuesta<br>– Revisión Documental | – Cuestionario<br>– Ficha Revisión Documental | – Estadística Descriptiva<br>– Análisis Documental | Documento con la caracterización de los elementos metodológicos del Centro de Informática de la Universidad de Nariño y los de Scrum |

Tabla 10. (Continuación)

| Objetivos específicos | Fuente | Técnica de recolección | Instrumento | Técnica de procesamiento | Resultado |
|---|---|---|---|---|---|
| Describir de manera comparativa los elementos metodológicos del proceso de construcción de software del centro de informática de la Universidad de Nariño y los de Scrum. | Documento con la caracterización de los elementos metodológicos del Centro de Informática de la Universidad de Nariño y los de Scrum. | – Revisión Documental | – Ficha Revisión Documental | – Análisis Documental | Matriz de comparación de los elementos metodológicos del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los de Scrum. |
| Formular una propuesta de trabajo basada en Scrum y apoyada por herramientas de software libre para el proceso de construcción de software del Centro de Informática de la | Matriz de comparación de los elementos metodológicos del proceso de construcción de software del Centro de Informática de la Universidad de | – Revisión Documental | – Ficha Revisión Documental | – Análisis Documental | Propuesta de trabajo basada en Scrum y apoyada por herramientas de software libre para el proceso de construcción de software |

Tabla 10. (Continuación)

| Objetivos específicos | Fuente | Técnica de recolección | Instrumento | Técnica de procesamiento | Resultado |
|---|---|---|---|---|---|
| Universidad de Nariño | Nariño y los de Scrum. | | | | del Centro de Informática de la Universidad de Nariño. |
| Aplicar la propuesta de trabajo basada en Scrum y apoyada por herramientas de software libre para el desarrollo de un producto software en el Centro de Informática de la Universidad de Nariño | Equipo de desarrollo del Centro de Informática de la Universidad de Nariño | – Encuesta | – Cuestionario | – Estadística Descriptiva | Documento con los resultados de la aplicación de la propuesta de trabajo basada en Scrum y apoyada por herramientas de software libre para el desarrollo de un producto software en el Centro de Informática de la |

Tabla 10. (Continuación)

| Objetivos específicos | Fuente | Técnica de recolección | Instrumento | Técnica de procesamiento | Resultado |
|---|---|---|---|---|---|
| | | | | | Universidad de Nariño |

Fuente: La presente investigación - 2017

### 1.7.5 Operacionalización de variables

**Variables:** En la siguiente tabla, se presentan las variables que se van a analizar en el estudio.

Tabla 11. Variables

| Variable | Descripción | Tipo de Variable | Objetivo específico | Indicador | Naturaleza | Fuente | Tr* | Ta** |
|---|---|---|---|---|---|---|---|---|
| Etapa | Etapa o fase del proceso para construir software | Independiente | - Caracterizar los elementos metodológicos, del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los definidos en el marco de trabajo Scrum.<br><br>- Describir de manera comparativa los elementos metodológicos del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los de Scrum. | – Etapa o fase del proceso de construir software realizado por el Centro de Informática de la Universidad de Nariño.<br>– Etapa o fase que plantea Scrum y se realiza en el Centro de Informática de la Universidad de Nariño. | Cualitativa | – Centro de Informática de la Universidad de Nariño.<br>– Metodología Scrum | – Encuesta<br>– Revisión documental | – Estadística descriptiva<br>– Análisis documental |

Tabla 11. (Continuación)

| Variable | Descripción | Tipo de Variable | Objetivo específico | Indicador | Naturaleza | Fuente | Tr* | Ta** |
|---|---|---|---|---|---|---|---|---|
| Actividad | Actividad que se desarrolla para alcanzar los objetivos del proyecto y producto | Independiente | - Caracterizar los elementos metodológicos, del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los definidos en el marco de trabajo Scrum.<br><br>- Describir de manera comparativa los elementos metodológicos del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los de Scrum | – Actividad que desarrolla el Centro de Informática de la Universidad de Nariño, para alcanzar los objetivos del proyecto y producto.<br>– Actividad que plantea Scrum y se realiza en el Centro de Informática de la Universidad de Nariño. | Cualitativa | - Centro de Informática de la Universidad de Nariño<br>– Metodología Scrum | – Encuesta<br>– Revisión documental | – Estadística descriptiva<br>– Análisis documental |
| Rol | Rol que desempeña la persona en el equipo de trabajo | Independiente | - Caracterizar los elementos metodológicos, del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los definidos en el marco de trabajo Scrum.<br><br>- Describir de manera comparativa los elementos metodológicos del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los de Scrum. | – Rol que desempeña la persona en el equipo de trabajo del Centro de Informática de la Universidad de Nariño.<br><br>– Rol que plantea Scrum y se ejecuta en el Centro de Informática de la Universidad de Nariño. | Cualitativa | – Centro de Informática de la Universidad de Nariño.<br><br>– Metodología Scrum | – Encuesta<br>– Revisión documental | – Estadística descriptiva<br>– Análisis documental |

Tabla 11. (Continuación)

| Variable | Descripción | Tipo de Variable | Objetivo específico | Indicador | Naturaleza | Fuente | Tr* | Ta** |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Artefacto | Entregable que sirve como documentación del proyecto y producto | Independiente | - Caracterizar los elementos metodológicos, del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los definidos en el marco de trabajo Scrum.<br><br>- Describir de manera comparativa los elementos metodológicos del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los de Scrum. | – Entregable que sirve como documentación del proyecto y producto en el Centro de Informática de la Universidad de Nariño.<br>– Entregable que plantea Scrum y se realiza en el Centro de Informática de la Universidad de Nariño. | Cualitativa | – Centro de Informática de la Universidad de Nariño.<br>– Metodología Scrum | – Encuesta<br>– Revisión documental | – Estadística descriptiva<br>– Análisis documental |
| Herramienta | Herramienta computacional de software libre, que sirve de soporte a la gestión del proceso de construcción de software | Independiente | - Caracterizar los elementos metodológicos, del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los definidos en el marco de trabajo Scrum.<br><br>- Describir de manera comparativa los elementos metodológicos del proceso de construcción de software del Centro de Informática de la | – Herramienta computacional de software libre, que sirve de soporte a la gestión del proceso de construcción de software en el Centro de Informática de la Universidad de Nariño.<br>– Herramienta computacional de software libre, que sirve de | Cualitativa | – Centro de Informática de la Universidad de Nariño.<br>– Metodología Scrum | – Encuesta<br>– Revisión documental | – Estadística descriptiva<br>– Análisis documental |

Tabla 11. (Continuación)

| Variable | Descripción | Tipo de Variable | Objetivo específico | Indicador | Naturaleza | Fuente | Tr* | Ta** |
|---|---|---|---|---|---|---|---|---|
| | | | Universidad de Nariño y los de Scrum. | soporte a Scrum y se utiliza en el Centro de Informática de la Universidad de Nariño. | | | | |
| Lineamiento | Principio que posibilita la toma de decisiones en el proceso de construcción de software | Independiente | - Caracterizar los elementos metodológicos, del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los definidos en el marco de trabajo Scrum.<br><br>- Describir de manera comparativa los elementos metodológicos del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los de Scrum. | – Principio que posibilita la toma de decisiones en el proceso de construcción de software en el Centro de Informática de la Universidad de Nariño.<br>– Principio de Scrum que posibilita la toma de decisiones en el Centro de Informática de la Universidad de Nariño. | Cualitativa | – Centro de Informática de la Universidad de Nariño.<br>– Metodología Scrum | – Encuesta<br>– Revisión documental | – Estadística descriptiva<br>– Análisis documental |

** Técnica de recolección **Técnica de análisis*

Fuente: La presente investigación - 2017

## 1.8. PRESUPUESTO

Tabla 12. Presupuesto materiales e insumos

| PRESUPUESTO MATERIALES E INSUMOS | | | |
|---|---|---|---|
| **CONCEPTO** | **CANTIDAD** | **VALOR UNITARIO** | **VALOR TOTAL** |
| Conexión a Internet (Mensual) | 6 | $50.000 | $300.000 |
| Resma de papel tamaño carta | 2 | $14.000 | $28.000 |
| Fotocopias | 300 | $50 | $15.000 |
| DVD en blanco | 3 | $1.200 | $3.600 |
| **Subtotal** | | | **$346.600** |

Fuente: La presente investigación 2017

Tabla 13. Presupuesto personal

| PRESUPUESTO PERSONAL | | | |
|---|---|---|---|
| **CONCEPTO** | **CANTIDAD** | **VALOR UNITARIO** | **VALOR TOTAL** |
| Director trabajo de grado (dedicación 2 horas semana) | 56 | $50.000 | $2.800.000 |
| Estudiante investigador (dedicación 15 horas semana) | 420 | $30.000 | $12.600.000 |
| **Subtotal** | | | **$15.400.000** |

Fuente: La presente investigación 2017

Tabla 14. Presupuesto de equipos

| PRESUPUESTO EQUIPOS | | | |
|---|---|---|---|
| **CONCEPTO** | **CANTIDAD** | **VALOR UNITARIO** | **VALOR TOTAL** |
| Computador | 1 | $2.500.000 | $2.500.000 |
| **Subtotal** | | | **$2.500.000** |

Fuente: La presente investigación 2017

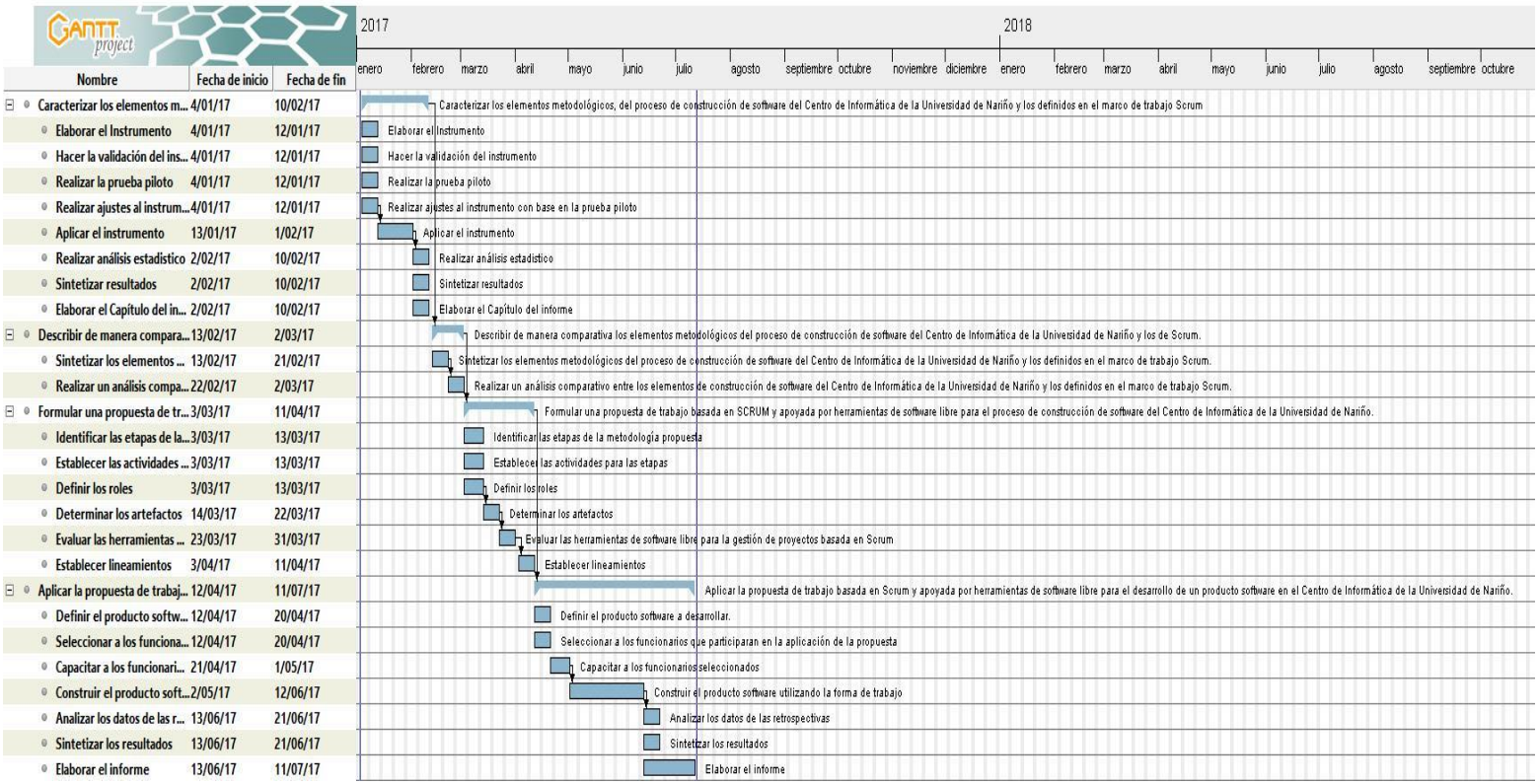
Tabla 15. Presupuesto general

| PRESUPUESTO GENERAL | |
|---|---|
| **CONCEPTO** | **VALOR** |
| Presupuesto materiales e insumos | $346.600 |
| Presupuesto Personal | $15.400.000 |
| Presupuesto Equipos | $2.500.000 |
| **TOTAL** | **$18.246.600** |

Fuente: La presente investigación 2017

## 1.9. CRONOGRAMA

Figura 7. Cronograma




Fuente: La presente investigación 2017

## 1.10. RESULTADOS ESPERADOS

Al finalizar este proyecto de investigación, se obtendrá como resultado:

- Propuesta de trabajo basada en Scrum y apoyada por herramientas de software libre para el proceso de construcción de software del Centro de Informática de la Universidad de Nariño.
- Informe con los resultados de la aplicación de la propuesta de trabajo basada en Scrum y apoyada por herramientas de software libre para el desarrollo de un producto software en el Centro de Informática de la Universidad de Nariño.
- Articulo resultado de la investigación.

# 2. RESULTADOS

## 2.1 CARACTERIZAR LOS ELEMENTOS METODOLÓGICOS, DEL PROCESO DE SOPORTE, MANTENIMIENTO Y CONSTRUCCIÓN DE SOFTWARE DEL CENTRO DE INFORMÁTICA DE LA UNIVERSIDAD DE NARIÑO Y LOS DEFINIDOS EN EL MARCO DE TRABAJO SCRUM

En esta sección se describe la caracterización de los elementos metodológicos para el proceso de soporte, mantenimiento y construcción de software del Centro de Informática de la Universidad de Nariño y los definidos en el marco de trabajo Scrum. Para alcanzar este objetivo, se tuvo como fuente de información al Centro de Informática y a los lineamientos definidos en Scrum. Las técnicas que se utilizaron para la recolección de información fueron: la encuesta (ver Anexo A) y la revisión documental. Para el análisis de la información, se utilizó como técnicas: la estadística descriptiva y el análisis documental. Las variables analizadas fueron: etapa, actividad, rol, artefacto, herramienta y lineamiento.

Una vez presentado el contexto y la forma como se realizó el análisis cuantitativo de los datos, se procede a mostrar los resultados encontrados. Los resultados que a continuación se presentan, describen inicialmente a la población de informantes. Luego, se detalla las percepciones de la población de acuerdo con las variables. Finalmente, se hace una síntesis de los resultados.

**2.1.1. Descripción sociodemográfica de la población.** La población encuestada estuvo conformada por 7 funcionarios del Centro de Informática de la Universidad de Nariño.

Como se puede observar en la Tabla 16, el 71.4% de los funcionarios encuestados pertenecen al género masculino y el 28.6% al género femenino.

Tabla 16. Distribución de funcionarios por género

| Categoría | FO - Frecuencia Observada | Frecuencia Observada (%) | FA - Frecuencia Acumulada |
|---|---|---|---|
| Masculino | 5 | 71,4 | 5 |
| Femenino | 2 | 28,6 | 7 |
| Total | 7 | 100 | |

Fuente: La presente investigación – 2017

En la tabla 17, se puede identificar que la mayoría de los funcionarios encuestados superan la edad de 26 años, lo cual indica que el personal que labora en el Centro de Informática tiene ya algunos años de experiencia en la profesión, que puede ser importante al momento de lograr los objetivos misionales de la dependencia.

Tabla 17. Distribución de funcionarios por edad

| Rango | FO - Frecuencia Observada | Frecuencia Observada (%) | FA - Frecuencia Acumulada |
|---|---|---|---|
| < 25 | 0 | 0 | 0 |
| 26 y 30 | 3 | 42,9 | 3 |
| 31 y 35 | 1 | 14,3 | 4 |
| > 35 | 3 | 42,9 | 7 |

Fuente: La presente investigación 2017

En la tabla 18, se indica el tiempo (años) que los funcionarios han trabajado en el Centro de Informática, se puede observar que la rotación del personal no es muy dinámica, pues la mayoría ya vienen laborando más de dos años en la dependencia, pero es importante aclarar que este es el grupo que permanece después de ser al menos doce empleados, los demás renunciaron por mejores condiciones laborales y sus funciones fueron asumidas por los que quedaban, de ahí la situación que se planteaba en la formulación del problema.

Tabla 18. Distribución de funcionarios por tiempo de trabajo.

| Rango (años) | FO - Frecuencia Observada | Frecuencia Observada (%) | FA - Frecuencia Acumulada |
|---|---|---|---|
| Menos de 1 año | 1 | 14,2 | 1 |
| Entre 1 y 2 años | 2 | 28,6 | 3 |
| Entre 2 y 3 años | 1 | 14,2 | 4 |
| Entre 3 y 5 años | 1 | 14,2 | 5 |
| Más de 5 años | 2 | 28,6 | 7 |

Fuente: La presente investigación 2017

En cuanto al cargo que desempeña cada uno de los funcionarios, las respuestas son muy ambiguas por ejemplo profesional de sistemas, profesional contratista, lo cual es un indicador del desconocimiento de las funciones específicas que cada uno debe desempeñar, traduciéndose en duplicación de actividades y desmejoramiento de la calidad de productos y servicios.

En la tabla 19. Se puede observar que el tiempo que los funcionarios llevan desempeñando el cargo, es el mismo que llevan trabajando en la dependencia, lo cual simplemente refuerza lo descrito en el punto anterior, en donde la falta de claridad en las funciones desempeñadas no permite establecer un grupo de cargos o roles lo cual dificulta a los funcionarios para identificarse con uno de ellos.

Tabla 19. Distribución de funcionarios por tiempo de desempeño en el cargo.

| Rango (años) | FO - Frecuencia Observada | Frecuencia Observada (%) | FA - Frecuencia Acumulada |
|---|---|---|---|
| Menos de 1 año | 1 | 14,2 | 1 |

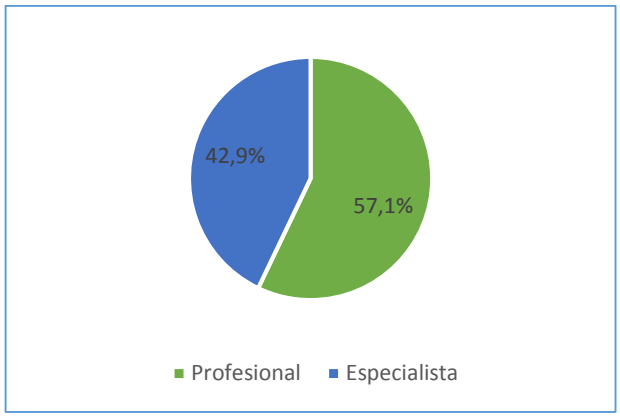
Tabla 19. (Continuación)

| Rango (años) | FO - Frecuencia Observada | Frecuencia Observada (%) | FA - Frecuencia Acumulada |
|---|---|---|---|
| Entre 1 y 2 años | 2 | 28,6 | 3 |
| Entre 2 y 3 años | 1 | 14,2 | 4 |
| Entre 3 y 5 años | 1 | 14,2 | 5 |
| Más de 5 años | 2 | 28,6 | 7 |

Fuente: La presente investigación 2017

En la figura 8, se puede observar que el Centro de Informática cuenta con un 57,1% de funcionarios con nivel máximo de formación profesional y un 42,9% con nivel de especialización. Lo cual es buen indicador del grado de preparación con el que cuenta el personal para asumir los diferentes retos de cara al avance tecnológico y también es una oportunidad para definir roles, aprovechando los conocimientos especializados o específicos adquiridos en los diferentes niveles de formación del personal.

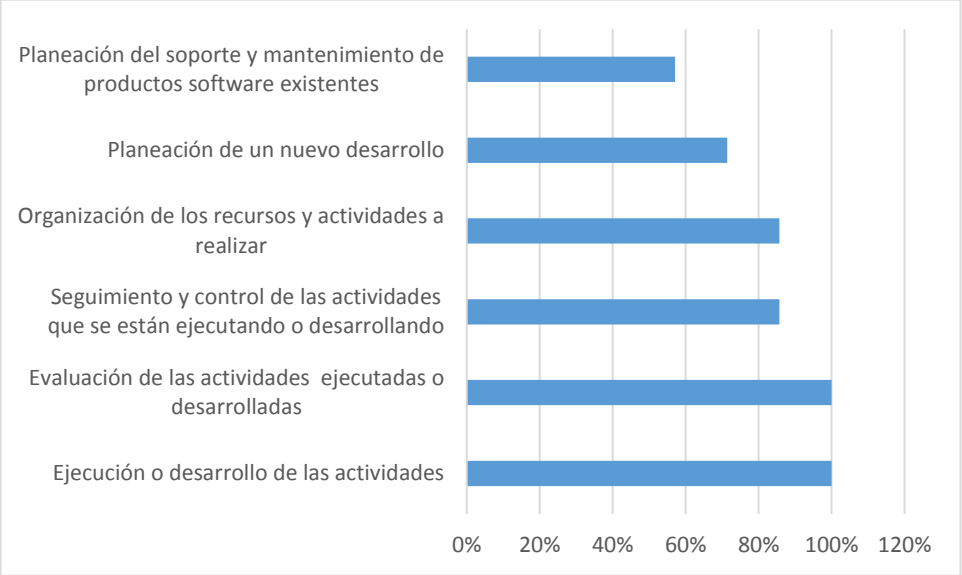
Figura 8. Distribución máximo nivel de formación.




Fuente: La presente investigación 2017

**2.1.2 Descripción de etapas o fases aplicadas en el proceso de desarrollo, soporte y mantenimiento de software.** Como es posible observar en la figura 9, la gran mayoría de fases o etapas se están aplicando, pero fácilmente se evidencia que fases tan importantes como la planeación tuvieron porcentajes más bajos en relación con las de ejecución, seguimiento y evaluación, muy posiblemente se deba a que los requerimientos de las diferentes unidades académico administrativas son urgentes, dando más énfasis a la fase de ejecución y poco espacio para la planeación. Lo fundamental sería que haya un equilibrio entre todas para afirmar que el trabajo realizado es estratégico y que nace de la aplicación de un proceso estandarizado.

Figura 9. Fases o etapas aplicadas para el desarrollo, soporte y mantenimiento de software.




Fuente: La presente investigación 2017

En cuanto al planteamiento de objetivos a alcanzar antes de iniciar el proceso de desarrollo de un nuevo producto, soporte y mantenimiento de software, todos los funcionarios del Centro de Informática indicaron que si se realiza, lo cual es contradictorio si se revisa el punto anterior, debido a que la fase de Planeación no la están realizando el 100% de ellos, y es justamente en esta fase donde se plantean los objetivos a alcanzar y se precisa las actividades necesarias para lograrlos.

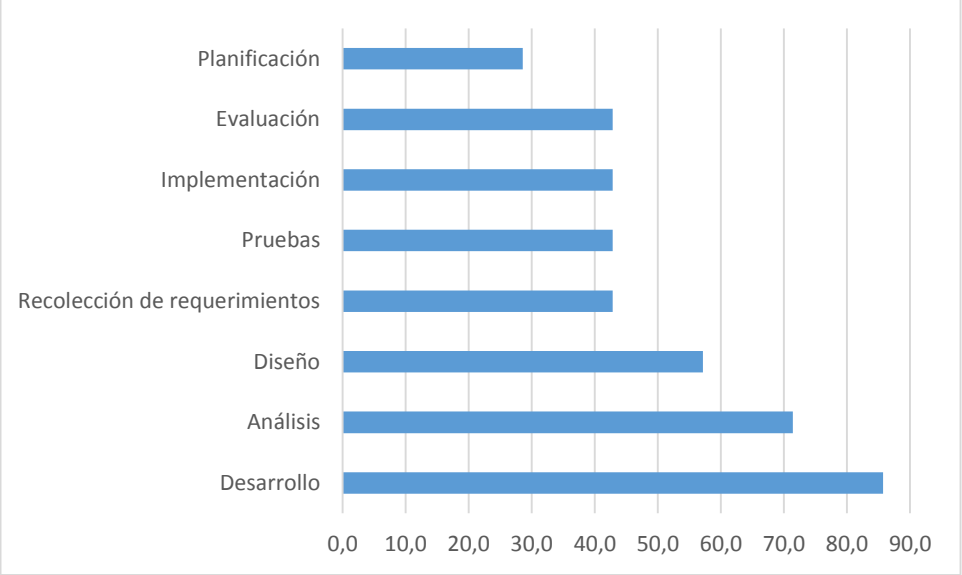
**2.1.3 Descripción de actividades aplicadas en el proceso de desarrollo, soporte y mantenimiento de software**. En la figura 12, se puede observar que el 70% de los funcionarios realizan Análisis, que es una actividad en la que generalmente se identifica el problema a resolver y se consideran las necesidades de los usuarios finales del software, para que éstas sean satisfechas identificando los módulos y funciones asociadas, la organización del código y los algoritmos a utilizar; para posteriormente iniciar con las actividades de desarrollo, que como algo particular tienen el porcentaje más alto con un 86%, indicando que se da más prioridad a ésta actividad, incluso que a la de Análisis de requerimientos, evidenciando la carencia del uso una metodología de gestión, en donde por el momento las actividades se ajustan a las circunstancias y prioridades del funcionario encargado.

Teniendo en cuenta a (Sommerville, 2011), la construcción de software no únicamente se debe visualizar como el desarrollo de las etapas del proceso de software (Análisis, diseño, codificación, pruebas, despliegue y mantenimiento); sino que además se debe gestionar el proceso de software. En este sentido la gestión implica que se debe planificar, organizar, ejecutar y evaluar; si se desea mejorar y asegurar la calidad de lo que se está realizando. Por lo tanto el proceso de software se convierte en parte de las actividades que se desarrollan en la etapa de ejecución.

Teniendo en cuenta los datos que se presentan en la figura 10, todas las actividades corresponden a la etapa de ejecución en la gestión del proceso software, a pesar de que en la indagación preliminar sobre las fases, el 100% indicó la aplicación de la fase de evaluación, se esperaría actividades propias de esta etapa, pero no se evidencia.

En cuanto a las actividades llevadas a cabo para realizar soporte y mantenimiento de software son variables para cada funcionario y no hay una estandarización de las mismas, como si fuese un proceso que se realiza en etapas y cada uno contribuye en una de ellas; desde el registro del soporte, pasando por la solicitud de documentación que lo sustenta, el análisis hasta la codificación de los nuevos requerimientos, que puede en algunos casos ser la corrección o ajustes al software en producción. Reflejando y reafirmando lo dicho por los propios funcionarios cuando se preguntó sobre la utilización de una forma o método de trabajo en el desarrollo de las diferentes actividades tanto de Soporte y mantenimiento como de realización de un nuevo producto software, a lo cual un 86% afirmo No y solo el 14%

*Figura 10. Distribución Actividades realización de un producto software.*




Fuente: La presente investigación 2017

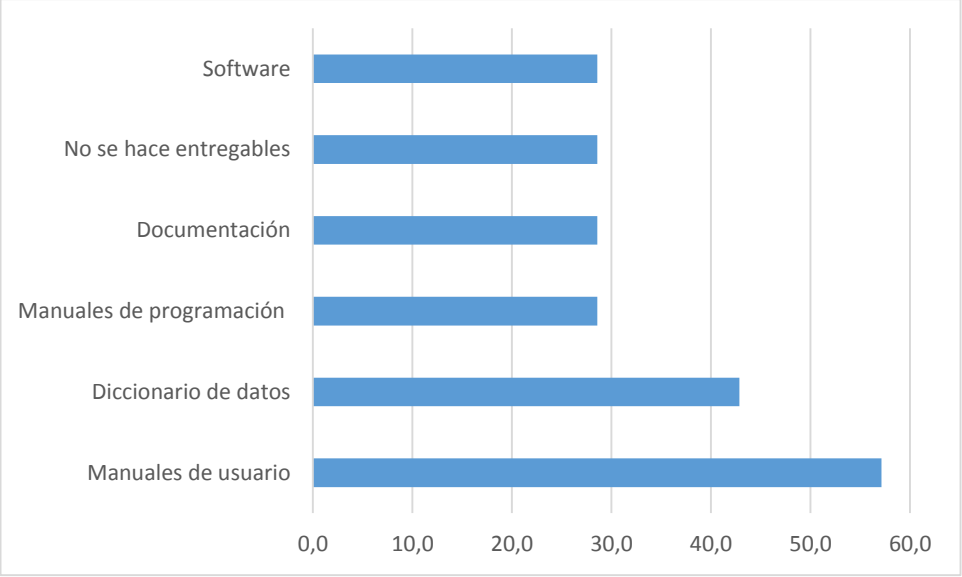
dijo Si; en donde se consideraría que cada integrante del Centro de Informática con el fin de cumplir las funciones designadas y las metas de la dependencia aplica las técnicas y actividades que según su criterio son las correctas.

**2.1.4 Descripción de roles dentro del proceso de desarrollo, soporte y mantenimiento de software.** Al realizar la aplicación de la encuesta un 57% de los encuestados afirmó que desempeña algún rol en las actividades de desarrollo de un nuevo producto, soporte o mantenimiento de software, el otro 43% restante que No. Para los funcionarios que respondieron de forma afirmativa a la existencia de roles, se logra identificar y corroborar que corresponden con las funciones que se cumplen en la ejecución, específicamente en funciones que se realizan en el proceso de software, como por ejemplo analista, diseñador y desarrollador. De ahí que contrastándolo con los análisis anteriores, es muy difícil hablar de definiciones de roles, ya que por lo general estos se identifican en la fase de Planeación, que para el caso del Centro de Informática por lo que se ha detectado, esta fase solo corresponde a la etapa de análisis en el proceso de desarrollo de software en la fase de ejecución y no a una verdadera fase de planificación en la gestión del software. Es importante indicar que cada funcionario está a cargo de un subsistema y es el responsable de su correcto funcionamiento, de ahí que en las operaciones

de desarrollo, soporte y mantenimiento se encargue de la elicitación de requerimientos, análisis, diseño de base datos, diseño de interfaces, codificación, pruebas y despliegue asumiendo siempre diversas funciones para cumplir los objetivos.

**2.1.5 Descripción de artefactos definidos en el proceso de desarrollo, soporte y mantenimiento de software.** En la figura 11, se puede observar que el entregable con mayor porcentaje cuando se desarrolla un nuevo producto software son los manuales de usuario, muy posiblemente debido a que son documentos que permitirán dar asistencia a los usuarios finales del software sobre su utilización y como llevar acabo las diferentes opciones disponibles, en segundo lugar con un 42% está el Diccionario de Datos que es un documento esencial que muestra todo el flujo y almacenamiento de datos que se manipula en el nuevo desarrollo, guardando descripciones y detalles de los mismos, otro de los entregables con un porcentaje importante son los manuales de programación, que en conjunto al Diccionario de datos, evitan que en futuros procesos de soporte y mantenimiento esta tarea sea compleja y tome demasiado tiempo.

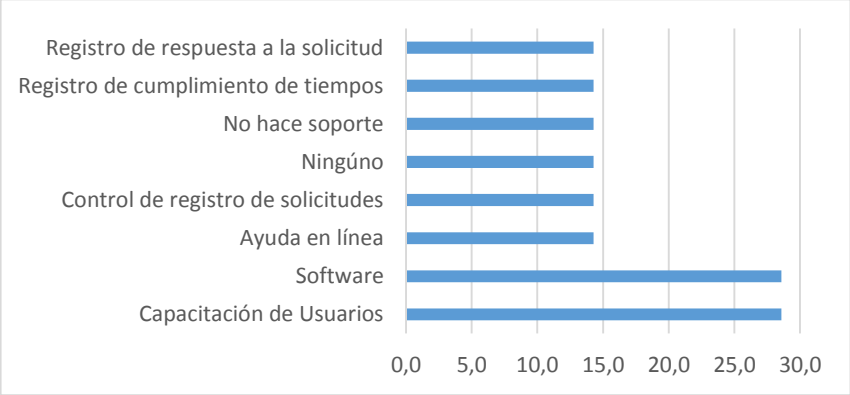
Figura 11. Distribución de Entregables nuevo producto Software.




Fuente: La presente investigación 2017

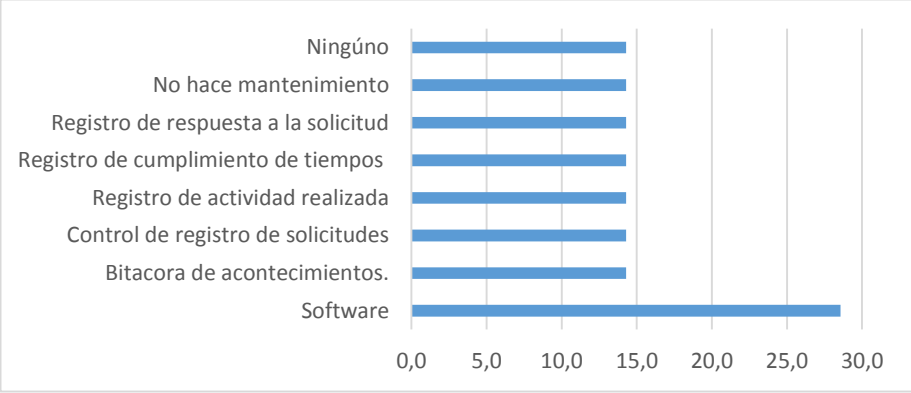
En las figuras 12 y 13, se evidencia que en el proceso de soporte y mantenimiento de los sistemas de información a cargo del Centro de Informática, los entregables generados son muy escasos y no existe unificación y estándar en los artefactos que se deben entregar, muy posiblemente debido a que los cambios se hacen sobre la marcha, y la urgencia de los mismos hace de que el personal no pueda dedicar tiempo suficiente a elaborarlos, solo a codificar las solicitudes o requerimientos y tratar de que funcionen rápidamente para ponerlos en producción y satisfacer las necesidades de los usuarios, de ahí que también sea necesario capacitarlos para que saquen el mayor provecho.

Figura 12. Distribución de Entregables en Soporte.




Fuente: La presente investigación 2017

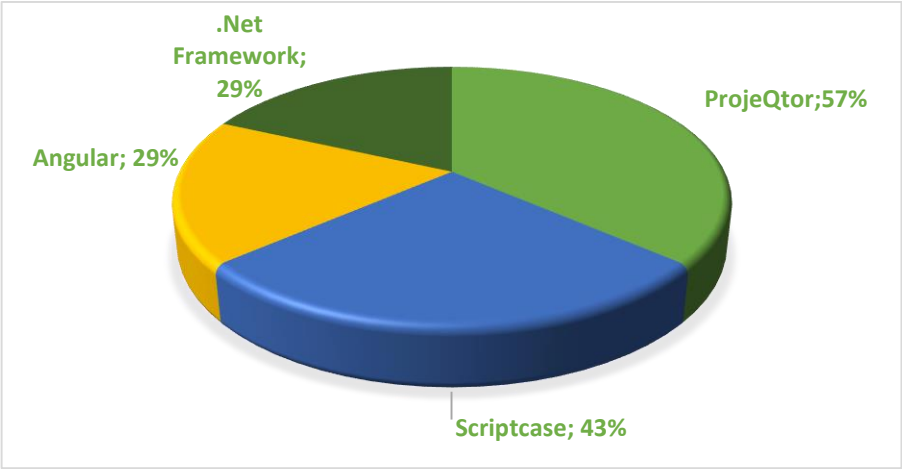
Figura 13. Distribución de Entregables en Mantenimiento.




Fuente: La presente investigación 2017

**2.1.6 Descripción de herramientas utilizadas como apoyo dentro del proceso de desarrollo, soporte y mantenimiento de software.** Observando la figura 14, una de las herramientas con el porcentaje más alto del 57% es ProjeQtor que es un software de código abierto para la administración de proyectos de software, utilizada por todos los funcionarios para el registro de los eventos diarios ejecutados en sus proyectos. En segundo lugar el framework Scriptcase basado en lenguaje PHP y utilizado para el desarrollo de aplicaciones en entorno web. Fue adquirido por el Centro de Informática con el objetivo de desarrollar formularios y reportes en un tiempo corto, debido a la dinámica de los requerimientos de la Universidad, para dar una mayor versatilidad en la entrega de los proyectos informáticos. Y en menor porcentaje dos *frameworks* muy potentes, .Net que es utilizado para el mantenimiento de sistemas heredados que fueron desarrollados en esta tecnología y Angular JS que hace parte de la nueva propuesta en tecnologías de desarrollo, que permitirá la creación de aplicaciones SPA (Single Page Aplication), que se comunicaran con los Web Services y transmitirán la información en formato JSON. Los funcionarios se están capacitando en su utilización y funcionamiento para que los nuevos desarrollos se realicen haciendo uso de este nuevo framework.

Figura 14. Distribución de Herramientas usadas en realización de un nuevo producto Software.




Fuente: La presente investigación 2017

Las herramientas usadas para la realización de soporte y mantenimiento de software son las mismas utilizadas para su creación, ya que los requerimientos de

las diferentes unidades académico administrativas involucran en algunos casos modificaciones de fondo, que demandan correcciones importantes al código, los datos y su estructura; en donde no solo se debe recurrir al framework, sino a programas especializados como EMS SQL que es una herramienta de administración grafica para bases de datos PostgreSQL, que hace que cualquier modificación o creación de bases de datos y sus componentes sea muy fácil de realizar. Adicional a lo anterior se evidencia la utilización de programas de ofimática como Excel y de diseño de interfaces de Usuario final.

**2.1.7 Descripción de lineamientos definidos en el proceso de desarrollo, soporte y mantenimiento de software.** Un punto importante sobre el que se indago fue si se hace uso de indicadores, factores o métricas dentro del proceso de desarrollo, soporte y mantenimiento de software por parte de los funcionarios del Centro de Informática, a lo cual un 71% dijo No, lo cual simplemente es un reflejo más de la falta clara de una metodología que contribuya a la gestión del software en todas sus etapas planeación, organización, ejecución y evaluación; en esta última la falta de recopilación de datos para establecer indicadores de desempeño en el desarrollo del software, hace que no se pueda llevar un control de los recursos, avance y ejecución de las diferentes tareas o actividades fijadas dentro de cada uno de los proyectos y por consiguiente el proceso de toma de decisiones no sea informado y bien estructurado, sino basado en la experiencia y en la percepción del director y grupo de trabajo del Centro.

**2.1.8 Caracterización de los elementos metodológicos del marco de trabajo Scrum.**

**2.1.8.1 Etapa.** Scrum maneja principalmente cuatro fases, las cuales se describe a continuación:

**Pre-Juego:** Se refiere a la planeación de cada iteración, incluye análisis y diseño. En esta fase el cliente debe presentar al equipo de trabajo una lista de requisitos priorizada del proyecto, se resuelven las dudas que puedan surgir y se hace una selección de los requisitos de forma prioritaria con los cuales se debe terminar la iteración que se está iniciando, es decir, se definen cuales requisitos se entregaran en la presente iteración.

Seguido a la definición de cuáles son los requisitos a entregar, el equipo se dedica a elaborar la lista de tareas correspondientes a la iteración, y que son necesarias para el desarrollo de los requisitos con los cuales se comprometieron. En este punto, la estimación se hace de forma conjunta y los propios miembros se auto-asignan sus tareas.

Es importante destacar que de esta fase surge una estimación de coste, cronograma y diseño de las funcionalidades. (SCRUM MANAGER, 2013).

**Juego:** Se deben realizar reuniones diarias de verificación y sincronización, con el fin de que los miembros del equipo puedan inspeccionar el trabajo de los demás miembros, en el caso que exista dependencias de tareas, y revisar el progreso para alcanzar el objetivo e inconvenientes que se están presentando, todo esto es con el fin de realizar adaptaciones que permitan más fácilmente cumplir con el compromiso que se adquirió en la fase anterior. En esta fase principalmente, se presenta el desarrollo de cada sprint, es decir el proceso de ir alcanzando el objetivo de la nueva funcionalidad.

**Post-Juego:** En esta última fase de la iteración se realiza una reunión de revisión consta de dos partes: Demostración, donde se presenta al cliente los requisitos implementados en la presente iteración (incremental) con el fin de definir resultados y cambios presentados en el proyecto que generan nuevas adaptaciones objetivas que permiten una re-planificación del proyecto. Retrospectiva, espacio donde el equipo debe analizar la forma en la que trabajó la presente iteración y debe también deducir cuales fueron los inconvenientes que no permiten generar un buen curso de trabajo, todo esto con el fin de mejorar continuamente la productividad, además el facilitador debe eliminar los obstáculos que se logran identificar. Es importante destacar que se debe entregar el incremento con la documentación y pruebas. (SCRUM MANAGER, 2013).

**Cierre:** Al momento de revisar que toda la versión ya está con los objetivos trazados, con los requisitos que se planearon tanto en coste como calidad, y que principalmente todo está alineado para producir una nueva versión, entonces se puede declarar la versión como cerrada.

**2.1.8.2 Actividad.** En cada una de las etapas indicadas anteriormente se realizan diferentes actividades que moldean la estructura del marco metodológico de Scrum. En la tabla 20, se listaran y explicará cada una de ellas.

Tabla 20. Actividades por Etapas

| ETAPA | ACTIVIDAD | DESCRIPCIÓN |
|---|---|---|
| Pre-Juego | Conformación del equipo Scrum. | Consiste en definir los integrantes de este equipo conformado por el Dueño del Producto, el Equipo de Desarrollo y el *Scrum Master*. Donde una característica principal es que son auto-organizados y multifuncionales, es decir que pueden elegir como hacer su trabajo sin tener la necesidad de depender de miembros externos al equipo, lo cual optimiza la creatividad, productividad y flexibilidad. |
| Pre-Juego | Definición de los roles | Esta actividad tiene como objetivo asignar los diferentes roles previamente identificados en la actividad de conformación del equipo Scrum.<br><br>***Product Owner*:** El Dueño del Producto se encarga de incrementar el valor del producto y el trabajo del equipo, y una de sus principales funciones es la gestión de la Pila de Producto (*Product Backlog*), para ello debe tener claridad en los elementos de la Pila, ordenar dichos elementos para lograr los objetivos; asegurar la calidad del trabajo del equipo; asegurar que la pila esté disponible y clara para todo el equipo; asegurar que el equipo entiende a un nivel necesario cada elemento de la Pila.<br><br>***Development Team*:** El equipo de desarrollo se compone por las personas encargadas de entregar cada incremento del producto, al final de cada sprint. Los miembros de este equipo tienen la independencia de organizar y gestionar su trabajo, incrementando de esta manera la eficiencia y efectividad. |

Tabla 20. (Continuación)

| ETAPA | ACTIVIDAD | DESCRIPCIÓN |
|---|---|---|
| Pre-Juego | Definición de los roles | ***Scrum Master*:** Es la persona encargada de que la metodología Scrum sea aplicada de forma correcta, ajustándose a la teoría, reglas y prácticas. También se encarga de informar a las personas externas al equipo que interacciones pueden ser útiles o no. |
| Pre-Juego | Instalación de las herramientas de soporte al proceso. | En la actualidad hay muchas herramientas software que contribuyen al uso de la metodología Scrum y su fácil gestión, por lo que es recomendable utilizar al menos una, de ahí la necesidad de realizar el proceso de instalación y configuración. |
| Pre-Juego | Definición del *done* | Cuando un elemento de la Lista de Producto o un Incremento se describa como "Terminado", todo el mundo debe entender lo que significa. De ahí que al definir el *done*, todos los miembros del equipo deben tener un entendimiento compartido de lo que significa que el trabajo esté completado, con el propósito de asegurar la transparencia y calidad de cada una de las iteraciones. |
| Pre-Juego | Definición de los artefactos en formatos estándar | La metodología Scrum, hace uso de dos artefactos Pila del producto (*Product Backlog*) y la Pila del sprint (*Sprint Backlog*) que son formatos para el registro de los requisitos y las historias de usuario. |
| Pre-Juego | Elaboración del *Product Backlog* | En esta actividad se debe listar los requisitos del sistema visto desde el punto de vista del cliente ordenados por la prioridad que el mismo da a cada uno. Esta lista nunca se da por terminada, siempre crece y evoluciona conforme avanza el desarrollo. |
| Pre-Juego | Definición del tiempo del *Sprint* | Todos los eventos son bloques de tiempo (time-boxes), de tal modo que todos tienen una duración máxima. Una vez que comienza un *Sprint*, su duración es fija y no puede acortarse o alargarse. Esta duración puede ser desde una, hasta seis semanas, aunque se recomienda que no exceda de un mes. |

Tabla 20. (Continuación)

| ETAPA | ACTIVIDAD | DESCRIPCIÓN |
|---|---|---|
| Juego | Priorización de los elementos del *Product Backlog* | Una vez elaborada la pila de producto resultado de reuniones en las que se aplica técnicas como "tormenta de ideas", "fertilización cruzada" o procesos de exploración (*eXtreme Programming*) en las que participan las personas que comparten la visión del propietario del producto, se procede a ordenar la pila por aquellos elementos más prioritarios, es decir los que confieren mayor valor al producto y cuyo desarrollo de sus actividades es inmediato; por consiguiente deben tener un nivel de detalle que permita descomponerlas en tareas, las cuales una vez ejecutadas den paso al siguiente *Sprint*. |
| Juego | Elaboración del *Sprint Backlog* | En esta actividad se realiza una lista de las tareas necesarias para construir las historias de usuario que se van a realizar en un sprint. |
| Juego | Ejecución de las tareas del *done* | En la fase de pre-juego se definió que tareas se deben realizar para dar por terminado un *Sprint* y es aquí donde se verifica que se ejecuten y sigan sus lineamientos. |
| Juego | Recopilación de datos para métricas ágiles | Es importante recopilar cierta información que será útil en el proceso de toma de decisiones como el Tiempo que falta para terminar, complejidad de las historias de usuario realizadas y la velocidad del equipo de trabajo. |
| Juego | Realización del *daily Scrum meeting* | Es necesario que el equipo de trabajo monitorice la evolución que se tiene en cada una de las tareas del *Sprint* y el trabajo pendiente, con la realización de reuniones muy cortas de 5 a 15 minutos máximo, en donde se revisa en conjunto el trabajo de cada uno de los miembros del día anterior y el previsto para el día actual. |
| Juego | Realización del *Sprint Review* | Al concluir el *Sprint* es necesario realizar una reunión para comprobar el incremento, su duración debe estar entre 1 y 4 horas máximo, en donde el dueño del producto evidencia el progreso del sistema e identifica las historias de usuario terminadas, las que faltan y lo más importante la visualización y prueba del |

Tabla 20. (Continuación)

| ETAPA | ACTIVIDAD | DESCRIPCIÓN |
|---|---|---|
| | | incremento, que permite tener retroalimentación relevante para revisar la pila de producto. |
| Post-Juego | Realización del *Sprint Retrospective* | Es importante realizar una reunión tras la revisión de cada *Sprint* y antes de planificar el siguiente, su duración puede ser de 1 a 3 horas en donde el equipo realiza un autoanálisis de cómo viene trabajando, permitiéndole identificar fortalezas que se deben mantener y debilidades que deben tener acciones de mejora. No se debe confundir con la anterior actividad ya que los objetivos de cada una son diferentes, en la anterior se analiza "QUE" se está construyendo, mientras que acá se enfoca en "COMO" se está construyendo con el objetivo de mejorar el marco de trabajo utilizado. |

Fuente: La presente investigación 2017

**2.1.8.3 Roles.** Dentro de la Metodología Scrum hay muchas personas que intervienen de forma directa o indirecta en el proyecto y se las clasifica en dos grupos: comprometidos e implicados. Dentro de los comprometidos se pueden identificar al propietario del producto, *Scrum master* y miembros del equipo y en los implicados a la dirección o gerencia. A continuación se hará una definición más detallada del grupo comprometidos.

***Product Owner*:** El Dueño del Producto es quien representa al cliente y toma sus decisiones, se encargará de incrementar el valor del producto y el trabajo del equipo, generalmente este rol debe recaer en una sola persona así el cliente sea una organización conformada por varios departamentos, porque dentro del equipo del proyecto solo se integra a un representante del cliente.

Dentro de las principales funciones de este rol está la de gestionar la Pila de Producto (*Product Backlog*) de la siguiente manera: Tener claridad en los elementos de la Pila; ordenar dichos elementos para lograr los objetivos; asegurar la calidad del trabajo del equipo; asegurar que la pila esté disponible y clara para todo el equipo; asegurar que el equipo entiende a un nivel necesario cada elemento de la Pila.

También debe decidir y conocer el plan del producto y cómo será el resultado final, el orden de desarrollo de los incrementos periódicos y la prioridad de las historias de usuario. Así como el retorno esperado a la inversión realizada, responsabilizándose sobre fechas de entrega de las versiones funcionales del producto.

La persona que vaya a desempeñar este rol debe conocer muy bien el entorno de negocio del cliente, los requerimientos y el objetivo a alcanzar con la ejecución del proyecto, para priorizar eficientemente el trabajo a desarrollar, de ahí que debe tener autonomía para tomar decisiones puesto que siempre recibirá retroinformación del negocio (movimiento del mercado, competencia, oportunidades) y del proyecto (recomendaciones del equipo, avances, pruebas y evaluación de cada incremento), la cual se debe analizar para trazar estrategias en pro del buen curso del proyecto.

***Development Team*:** El equipo de desarrollo se compone por las personas encargadas de entregar cada incremento del producto, al final de cada sprint. Los expertos recomiendan que el número este entre no menos de 3 ni más de 9 integrantes, ya que esto puede dificultar la comunicación directa. Dentro de este equipo no se deben incluir al Scrum Master ni el propietario del producto. Se debe tener claro que no es un grupo de trabajo en cual simplemente se asignan funciones, es un equipo multifuncional en el que por supuesto hay miembros con especialidades concretas, pero el trabajo es solidario y la responsabilidad compartida con un propósito común: lograr el incremento de cada *Sprint* y conseguir el mayor valor posible para la visión del cliente. Para ello es necesario que todos conozcan y comprendan la visión del propietario del producto y colaboren en el desarrollo de la pila del producto y compartan el objetivo de cada *Sprint* y la responsabilidad de alcanzarlo, lo anterior es posible ya que afortunadamente los miembros de este equipo tienen la independencia de organizar y gestionar su trabajo, respetando las opiniones y aportes de todos incrementando de esta manera la eficiencia y efectividad de su trabajo.

***Scrum Master*:** Es la persona encargada de que la metodología Scrum sea aplicada de forma correcta ajustándose a la teoría, reglas y prácticas, asegurando que se entienden en la organización y se trabaje acorde a ellas. Este rol desempeña diversas responsabilidades entre ellas asesorar y formar al equipo para que cumpla su misión, también debe revisar y validar constantemente la pila de producto para

que haya una lista priorizada de requisitos antes de la siguiente iteración. En las diferentes reuniones (Planificación de la iteración, reuniones diarias de sincronización del equipo, demostración, retrospectiva) debe hacer las veces de moderador de manera que estas sean productivas y logren su propósito. De igual forma enseña al equipo auto gestionarse en la resolución de dificultades en el *Sprint* o en la dinámica de grupo, que puedan afectar el desarrollo normal de las tareas y por su puesto una tarea muy importante es que está mejorando continuamente las prácticas de Scrum en la organización.

**2.1.8.4 Artefactos.** La metodología Scrum propone dos herramientas o "Artefactos" que contribuyen a la planificación y maximización de la transparencia de información clave en la toma de decisiones, ya que aseguran que todos tengan la misma comprensión del artefacto. Si este no es lo suficientemente transparente, las decisiones que se tomen pueden ser erróneas, haciendo que el valor del producto disminuya y el riesgo se incremente.

***Product Backlog*:** Es una lista que contiene todos los requerimientos vistos desde el punto de vista del cliente para las funcionalidades del producto esperado, debido a su carácter dinámico esta lista nunca se da por terminada, siempre crece y evoluciona con base en el feedback realizado en cada sprint o a cambios de requisitos del negocio, tendencias del mercado o de tecnología.

El único responsable del contenido, ordenación y disponibilidad de la lista es el *Product Owner*, que para el caso de la ordenación se tiene en cuenta la prioridad de los elementos, siendo más prioritarios los que dan mayor valor al producto o cuyo desarrollo es inmediato y por consiguiente deben tener un alto nivel de detalle que permita descomponerlos en tareas.

En cuanto al formato utilizado como ya se ha mencionado se hace uso de una lista, en donde se indican elementos como:

- Código o identificador único de historia de usuario
- Descripción de la funcionalidad/requisito, denominado "historia de usuario".
- Prioridad
- Preestimación del esfuerzo necesario

Dependiendo de las características del proyecto o preferencias del equipo se pude incluir información adicional como observaciones, criterios de validación, número de *Sprint* en el que se realiza etc.

Para el atributo estimación el responsable de definirlo es el equipo de desarrollo (*Development team*), ya que son los encargados de hacer el trabajo y quienes se comprometen a lograr el objetivo planteado para el *Sprint*, en cuanto más detallados y claros estén los elementos, más precisa será su estimación.

***Sprint Backlog***: Es un subconjunto de elementos de la pila de producto (*Product Backlog*) seleccionados por el equipo en la reunión de planificación del *Sprint* en donde se asigna a cada tarea el esfuerzo previsto para realizarla y que debe cumplirse para hacer la entrega del incremento terminado, por ende debe ser tan explícito que permita visualizar las acciones a seguir para lograr el objetivo del *Sprint*, al igual que la pila de producto es un artefacto dinámico que puede tener cambios y a medida que se va trabajando se van actualizando por ejemplo atributos como la estimación del trabajo restante. Para facilitar su gestión generalmente se hace uso de un tablero físico, hojas de cálculo o herramientas específicamente diseñadas para trabajar con la metodología, que indicarían información como la pila del *Sprint*, la persona responsable de cada tarea, su estado y tiempo que falta para completarla. Se debe estar actualizando cada día durante el *Sprint*, también con estos datos se traza el grafico conocido como Burn-Down del cual se habla a continuación en Métricas Ágiles.

**2.1.8.5 Lineamientos.** En la gestión de proyectos la medición del trabajo no es una tarea sencilla, se debe determinar que medir, en donde una decisión equivocada, en el mejor de los casos solo supondrá costos que pudieron ser evitables, pero también puede agravar lo que se trataba de mejorar; de ahí que los expertos indican que antes de incorporar un procedimiento de medición, se debe discutir cual será el beneficio o valor que proporcionará y la forma en que se aplicará.

En la metodología Scrum las métricas ágiles son el elemento que permite recopilar datos y convertirlos en información, como materia prima para la toma de decisiones, estas permiten definir que medir y establecer la importancia y la forma como se realizará la medición. Por ejemplo, se puede medir valor de las tareas en desarrollo o el trabajo pendiente por realizar. Como el objetivo de Scrum es producir el mayor

valor posible de forma continua, una de las variables a medir es la velocidad que se define como la cantidad de trabajo realizada por el equipo en un *Sprint*.

Para mantener un ritmo de progreso constante, el desarrollo ágil utiliza dos técnicas el incremento iterativo y el incremento continuo, el primero es el usado por Scrum técnico en donde el avance es a través de incrementos iterativos apoyados por Sprints de ahí que se los emplee como unidad de tiempo. En lo relacionado a la medición del trabajo con el objetivo de determinar el grado de avance del proyecto, puede ser importante medir el trabajo ya realizado, lo cual no implica mucho esfuerzo, puesto que se cuenta lo desarrollado en la unidad empleada que pueden ser horas trabajadas, líneas de código etc. Pero para el caso del desarrollo ágil este objetivo se logra midiendo el trabajo pendiente por realizar, para de esta manera estimar el esfuerzo y tiempo necesario en la realización de las tareas o historias de usuario; lo cual es algo que no se puede predecir de forma absoluta, ya que entran en juego muchas variables sobre las que no se tiene un verdadero control, de ahí que en la gestión ágil se haga uso de estrategias como: trabajar con estimaciones aproximadas, utilizar la técnica de “Juicio de Expertos” y dividir las tareas en subtareas más pequeñas. La unidad de trabajo para estas estimaciones son los “puntos”, también llamados “Puntos de Historia” en donde cada organización determina la correlación de su punto con su métrica de trabajo, donde es importante que el significado y como se aplica sea una constante en todas las mediciones, que se institucionalice y se conozca por todas las personas.

### 2.1.9 Síntesis de Resultados.

El cuestionario aplicado a los funcionarios del Centro de Informática se estructuro de tal forma que las preguntas permitieran vislumbrar como era la forma de trabajo de esta dependencia dentro del proceso de soporte, mantenimiento y construcción de software, en donde las respuestas fueron muy valiosas y permitieron hacer un análisis de las variables etapa, actividad, rol, artefacto, herramienta y lineamiento dentro del marco de trabajo. Permitiendo determinar que para el caso de las etapas la mayoría están enmarcadas dentro del proceso de ejecución y se descuidan las de planeación, organización y evaluación correspondientes a la gestión de software, en cuanto a las actividades siguen el mismo patrón y se enfocan en la obtención rápida de los requerimientos de los usuarios, para continuar con su codificación y una vez terminado el producto integrarlo al sistema en producción muchas veces sin la suficientes pruebas y la generación de documentos que soporten el proceso realizado. Tampoco hay una definición clara de los roles en la dependencia, simplemente todos desempeñan funciones comunes a la ingeniería de sistemas,

que no se ajustan a un cargo especifico, generando muchas veces duplicidad y desgaste administrativo; por otra parte al no realizar una verdadera gestión del proceso de desarrollo de software, cada quien es su propio líder, ejecutor y evaluador, dificultando la generación estandarizada de entregables o artefactos, lo cual se pudo evidenciar claramente al indagar sobre este punto, en donde nuevamente los pocos artefactos generados se enmarcan en la fase de ejecución y a criterio del funcionario.

En la investigación se identificó el uso de herramientas como frameworks, gestores de bases de datos, editores de texto entre otros y en menor grado aquellas que contribuyen con la gestión de proyectos de software en todas sus fases, haciendo difícil el seguimiento a variables tan importantes como el tiempo, recursos y manejo de indicadores para la toma de decisiones basadas en criterios reales y medibles. De ahí que al indagar sobre el uso de métricas dentro del proceso de desarrollo, soporte y mantenimiento de software se pudo constatar que No se manejan, posiblemente debido a que en dicho proceso no se almacena información relevante que permita establecer indicadores, para posteriormente analizarlos y así determinar acciones en pro de mejorar continuamente la calidad de los productos y servicios ofrecidos por la dependencia.

## 2.2 ANÁLISIS COMPARATIVO DE LOS ELEMENTOS METODOLÓGICOS DEL PROCESO DE CONSTRUCCIÓN DE SOFTWARE DEL CENTRO DE INFORMÁTICA DE LA UNIVERSIDAD DE NARIÑO Y LOS DE SCRUM.

En esta sección se describe de manera comparativa los elementos metodológicos del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los de Scrum. Para alcanzar este objetivo, se tuvo como fuente de información el documento con la caracterización de los elementos metodológicos del Centro de Informática y los de Scrum. La técnica que se utilizó para la recolección de información fue la revisión documental. Para el análisis de la información se utilizó como técnica el análisis documental. Las variables analizadas fueron: Etapa, Actividad, Rol, Artefacto, Herramienta y Lineamiento.

Una vez presentado el contexto y la forma como se realizó el análisis cuantitativo de los datos, se procede a mostrar los resultados encontrados.

### 2.2.1 Etapa

A continuación se presenta la Tabla 21, que muestra el análisis comparativo realizado entre el elemento metodológico etapa y Scrum.

Tabla 21. Análisis comparativo para etapa

| Síntesis Scrum | Síntesis Centro de Informática Universidad de Nariño | Similitudes | Diferencias | Elementos de Intervención |
|---|---|---|---|---|
| Pre-juego:<br>• Planeación de la iteración<br>• Lista priorizada de requisitos<br>• Estimación de Coste, cronograma y funcionalidades | • Lista de requisitos<br>• Análisis y diseño de los elementos necesarios para la codificación. | En esta etapa en ambos casos se cuenta con una lista de requisitos o requerimientos sobre los cuales se orienta el proceso, ya sea de desarrollo, soporte o mantenimiento del software. | Realizando un análisis se puede identificar claramente que una vez se tienen los requerimientos del cliente, simplemente estos se codifican sin realizar una verdadera planificación del trabajo, sin estimación de tiempos de las tareas a realizar a través de un cronograma y de los costos asociados. | Es necesario dar más importancia a la fase de planeación y organización, ya que en el momento todo el trabajo se centra en la fase de ejecución; por consiguiente, es muy difícil hacer retroalimentación y evaluación de las actividades para mejorar la calidad de los productos software. |
| Juego:<br>• Reuniones diarias de sincronización y verificación<br>• Desarrollo del *Sprint* | • Codificación de los requerimientos previamente identificados<br>• Pruebas de software | La etapa de juego planteada en Scrum se desarrolla parcialmente por el Centro de Informática, ya que se lleva a cabo la codificación del software y | En el Centro de Informática se lleva a cabo la codificación del software total, mientras que en Scrum se lleva a cabo por cada *Sprint* entregando una versión funcional. | Definir tiempo para reuniones diarias con el fin de revisar el avance, dificultad y valor que agrega cada actividad propuesta. |

Tabla 21. (Continuación)

| Síntesis Scrum | Síntesis Centro de Informática Universidad de Nariño | Similitudes | Diferencias | Elementos de Intervención |
|---|---|---|---|---|
| | | pruebas como actividades que se desarrolla en cada *Sprint*, mediante tareas del *done*. | | |
| Post-juego:<br>• Entrega prototipo funcional.<br>• Retrospectiva (Análisis de la forma de trabajo) | • Publicación del nuevo desarrollo o soporte y mantenimiento realizado al software, en la plataforma Tecnológica de la Universidad. | No se encontró similitudes | En el Centro de Informática se realiza la publicación total del software, sin entregas parciales, a diferencia de Scrum donde se hacen entregas incrementales. Tampoco se hace una retrospectiva de las posibles dificultades encontradas en el trabajo realizado con el objetivo de tomar correctivos y mejorar la productividad | Incorporar instrumentos de inspección de artefactos.<br><br>Desarrollar o mantener un producto software de forma iterativa e incremental.<br><br>La generación de la documentación sea automática y se extienda no únicamente a manuales.<br><br>Retroalimentación en cuanto a inconvenientes dentro de la ejecución del *Sprint*. |

Fuente: La presente investigación 2017

**2.2.2 Actividad.** A continuación, se presenta la Tabla 22, que muestra el análisis comparativo realizado entre el elemento metodológico actividad y la metodología Scrum.

Tabla 22. Análisis comparativo para Actividad

| Síntesis Scrum | Síntesis Centro de Informática Universidad de Nariño | Similitudes | Diferencias | Elementos de Intervención |
|---|---|---|---|---|
| Pre-juego: | • Listar requerimientos | No son muy marcadas y las | Las actividades desarrolladas | • Conformación equipo Scrum |
| Pre-juego:<br>• Conformación equipo Scrum<br>• Definición de roles<br>• Instalación herramientas de soporte<br>• Definición del Done<br>• Definición de Artefactos<br>• Elaboración del *Product Backlog*<br>• Definición del tiempo del *Sprint* | • Asignación de responsable<br>• Comunicación con Usuarios o clientes | únicas que se podrían asociar serian la definición del listado de requerimientos por parte de los funcionarios para el desarrollo, soporte y mantenimiento del software, con la definición de los artefactos propuesto por Scrum. | por el Centro de Informática giran en torno a la fase de ejecución, donde el objetivo es definir rápidamente los elementos a necesitar, ya sea mediante comunicación con el cliente o a través de revisión documental para así dar inicio a la codificación. A diferencia de Scrum en donde las actividades se orientan a la planificación del proceso, para luego pasar a la fase de ejecución, controlando mejor los posibles sucesos e incertidumbres. | • Definición de roles<br>• Instalación herramientas de soporte<br>• Definición del Done<br>• Definición de Artefactos<br>• Elaboración del *Product Backlog*<br>• Definición del tiempo del *Sprint* |

Tabla 22. (Continuación)

| | Síntesis Centro de Informática Universidad de Nariño | Similitudes | Diferencias | Elementos de Intervención |
|---|---|---|---|---|
| | • Diseño de base de datos e interfaces<br>• Codificación<br>• Implementar el software. | No se encontró ninguna | En cuanto a las actividades realizadas en la etapa del juego las diferencias son bastante notables iniciando porque no se definen espacios de tiempo para el trabajo con los requerimientos como es el caso del *Sprint Backlog*. También como no hay definición previa de cuando un producto deba considerarse por terminado, entonces simplemente si el producto es funcional con eso basta y como no se usan métricas no hay necesidad de recopilar datos para utilizarlas. Debido a que en el C.I. por lo general a un proyecto se asigna solo un funcionario, no se realizan reuniones diarias para determinar avances y dificultades en | Definición clara de iteraciones, con los objetivos que se deben alcanzar.<br><br>Definir mecanismos de revisión y aplicación de estándares.<br><br>Definir tiempo para reuniones diarias de verificación de avances. |

Tabla 22. (Continuación)

| | Síntesis Centro de Informática Universidad de Nariño | Similitudes | Diferencias | Elementos de Intervención |
|---|---|---|---|---|
| | | | el desarrollo de los requerimientos y finalmente después de pruebas se verifica el funcionamiento del software y se publica; en caso de posibles errores notificados por los usuarios, nuevamente estos se analizan y se procede a corregirlos. | |
| | | No se encontró ninguna | No se encontró ninguna | • Realización del *sprint retrospective* |

Fuente: La presente investigación 2017

**2.2.3 Rol.** A continuación, se presenta la Tabla 23, que muestra el análisis comparativo realizado entre el elemento metodológico Rol y la metodología Scrum.

Tabla 23. Análisis comparativo para Rol

| Síntesis Scrum | Síntesis Centro de Informática Universidad de Nariño | Similitudes | Diferencias | Elementos de Intervención |
|---|---|---|---|---|
| Pre-Juego:<br>• *Product Owner* | • Desarrollador | En el centro de informática el único rol visible es el desarrollador | En el Centro de Informática no se define el perfil de Scrum Master y | Definir funciones claras para cada rol. |

Tabla 23. (Continuación)

| Síntesis Scrum | Síntesis Centro de Informática Universidad de Nariño | Similitudes | Diferencias | Elementos de Intervención |
|---|---|---|---|---|
| Pre-Juego:<br>• Scrum Master<br>• *Development Team* | • Desarrollador | como un integrante del *Development team*. | *Product Owner*, prioritarios dentro de la metodología Scrum. | Involucrar totalmente al cliente dentro del desarrollo, soporte y mantenimiento del software que compone el Sistema de Información Integrado de la Universidad, formando parte activa del Equipo Scrum. |

Fuente: La presente investigación 2017

**2.2.4 Artefactos.** A continuación, se presenta la Tabla 24, que muestra el análisis comparativo realizado entre el elemento metodológico Artefacto y la metodología Scrum.

Tabla 24. Análisis comparativo para Artefacto.

| Síntesis Scrum | Síntesis Centro de Informática Universidad de Nariño | Similitudes | Diferencias | Elementos de Intervención |
|---|---|---|---|---|
| Pre-juego:<br>• La pila del producto (*Product backlog*).<br>• La pila sprint | • Elicitación de Requisitos.<br>• Estructura de Base de Datos | Uno de los artefactos en común es la identificación de los requerimientos que para el caso del C.I. se define con prioridad | En el C.I. no se define el *Sprint backlog* que se refiere a las historias de usuario que mayor valor agregan al proceso del | Definir claramente dos artefactos primordiales de Scrum, el *Product Backlog* y el *Sprint Backlog*. |

Tabla 24. (Continuación)

| Síntesis Scrum | Síntesis Centro de Informática Universidad de Nariño | Similitudes | Diferencias | Elementos de Intervención |
|---|---|---|---|---|
| Pre-juego: | | para iniciar con su codificación, de la misma forma en Scrum se crea el *Product Backlog* que agrupa los requisitos del sistema, para luego en cada *Sprint* definir cuales se van a llevar a la fase de desarrollo. | cliente, y de un objetivo que se logrará en la entrega de incrementos del producto. | |
| Post-juego:<br>• Instrumento para especificar Métricas ágiles | • Código fuente.<br>• Documentación (Manuales de Usuario).<br>• Capacitaciones | Se puede determinar algunas analogías en el sentido de que cada incremento realizado en Scrum, está compuesto por el código fuente, la respectiva documentación, pruebas etc, todo lo que se haya definido en el Done, para el caso del C.I. se definen parcialmente algunos de estos elementos. | En Scrum en la definición del Done, se especifica que actividades se deben realizar antes de dar por terminado el trabajo y hacer la entrega del "incremento" el cual es el prototipo funcional de cada *Sprint* finalizado. | Es necesario construir el instrumento para la gestión de las métricas como:<br>• Tiempo que resta.<br>• Velocidad de Trabajo.<br>• Complejidad. |

Fuente: La presente investigación 2017

### 2.2.5 Lineamiento

A continuación, se presenta la Tabla 25, que muestra el análisis comparativo realizado entre el elemento metodológico Lineamiento y la metodología Scrum.

Tabla 25. Análisis comparativo para Lineamiento.

| Síntesis Scrum | Síntesis Centro de Informática Universidad de Nariño | Similitudes | Diferencias | Elementos de Intervención |
|---|---|---|---|---|
| Indicadores:<br>• Tipo de Trabajo por realizar.<br>• Velocidad de trabajo.<br>• Complejidad. | No se recopilan datos, ni se definen indicadores. | No hay ninguna. | No se recopila datos sobre trabajo realizado o por realizar, velocidad y complejidad. | • Tipo de Trabajo por realizar.<br>• Velocidad de trabajo.<br>• Complejidad. |

Fuente: La presente investigación 2017

### 2.2.6. Síntesis de Resultados

Al realizar la comparativa de los elementos metodológicos entre el Centro de Informática y Scrum se pudo evidenciar que son muy pocos los puntos en los que hay una similitud y al contrario existen muchas diferencias, para el caso de la variable etapa es común al inicio trabajar en la especificación de los requisitos del software en la cual los usuarios y desarrolladores definen el producto software a producir. Posteriormente se realiza la codificación pero a diferencia de Scrum no se hacen entregas parciales del producto conforme se avanza, sino que una vez finaliza el desarrollo y se aplican pruebas se despliega y presenta al cliente. Tampoco se hace retroalimentación en el proceso o lo que se conoce en Scrum como retrospectiva, cuyo objetivo es hacer un autoanálisis de cómo se está trabajando identificando fortalezas y debilidades para tomar correctivos a tiempo.

Para la variable Actividad la única similitud encontrada es la definición de requerimientos que en Scrum se denomina como el Producto *Backlog*, por lo demás el trabajo se centra en actividades propias de la fase de ejecución como diseño de base de datos, de interfaces, codificación y despliegue; a diferencia de Scrum donde

el proceso se enfoca en actividades que permitan planificar y organizar el trabajo tales como definición de roles, de tareas, elaboración de artefactos como el sprint *backlog* y la estimación en tiempo de cada requerimiento o historia de usuario.

Para la variable rol la única similitud asociada con Scrum es el rol del *Development team*, ya que en el Centro de Informática no hay una definición clara de los roles y los funcionarios son multifuncionales.

Para la variable artefacto la analogía más clara con Scrum es la del *Product Backlog*, con la lista de requerimientos que el funcionario del Centro obtiene a través de reuniones con el cliente o revisión documental, en lo relacionado al artefacto Métricas de Scrum, no se evidencia su definición, que permita evaluar el proceso de creación, soporte y mantenimiento del software.

Y en lo relacionado a la variable Lineamiento no hay similitudes, ya que a diferencia de Scrum no se recopilan datos, ni se definen métricas que permitan evaluar el desarrollo, soporte y mantenimiento del software a cargo de la dependencia.

## 2.3 SELECCIÓN DE LA HERRAMIENTA DE SOFTWARE LIBRE PARA LA GESTIÓN DEL PROCESO SCRUM.

Para determinar cuál de las herramientas de software libre para la gestión de Scrum se implementará en el Centro de Informática, inicialmente se realizó la evaluación de algunas de ellas utilizando el modelo OAM-F/OSS (*open appraisal model for free and open source software*), que es un modelo abierto que está conformado por cuatro fases: planeación, ejecución, verificación y selección.

El modelo plantea unos requisitos iniciales que debe tener en cuenta el evaluador como contar con los manuales de usuario e instalación de las herramientas, instalar las herramientas en entornos de prueba utilizando sistemas anfitriones o máquinas virtuales y tener comunicación con los funcionarios encargados del proceso que la herramienta pretende sistematizar con el objetivo de obtener información.

A continuación se explica en que consiste cada una de las fases del modelo:

**Fase 1 - Planeación:** Inicialmente se debe realizar la ponderación de cada uno de los nueve criterios que plantea el modelo (Aceptación / Usabilidad, Administración, Comunidad, Eficiencia, Entrenamiento, Integración, Portabilidad, Software / Producto, Especificidad). Posteriormente el evaluador debe establecer por cada uno el nivel de importancia en una escala del 1% al 100% como se indica en la tabla 26.

Tabla 26. Criterios de evaluación.

| Criterio | Ponderación |
|---|---|
| Aceptación / Usabilidad | % |
| Administración | % |
| Comunidad | % |
| Eficiencia | % |
| Entrenamiento (Capacitación/Documentación) | % |
| Integración | % |
| Portabilidad | % |
| Software / Producto | % |
| Especificidad | % |
| **Total** | **100%** |

Fuente: Modelo de evaluación para la selección de herramientas de software libre en el proceso de gestión documental. (Jiménez, 2016).

Posteriormente se debe operacionalizar las características a evaluar, el modelo propone una escala para la evaluación como se indica en la tabla 27.

Tabla 27. Escala de Evaluación

| Valoración | Juicio |
|---|---|
| **5** | Totalmente de acuerdo |
| **4** | De acuerdo |
| **3** | Ni de acuerdo, ni en desacuerdo |
| **2** | En desacuerdo |
| **1** | Totalmente en desacuerdo |

Fuente: Modelo de evaluación para la selección de herramientas de software libre en el proceso de gestión documental. (Jiménez, 2016)

El modelo también define las características a evaluar para cada uno de los criterios como se indica en la tabla 28, en donde el evaluador tiene la libertad de seleccionar las que crea conveniente y también adicionar nuevas dependiendo de la herramienta a evaluar, estas se asociarán al criterio de especificidad.

Tabla 28. Características de Evaluación

| Criterio | Característica | Seleccionada |
|---|---|---|
| Aceptación / Usabilidad | Usabilidad | |
| | Penetración en el mercado | |
| | Interfaz de usuario | |
| | Mantenibilidad | |
| | Implementación | |
| | Facilidad de despliegue | |
| Administración | Pruebas | |
| | Actividad de versiones | |
| | Actividad en características | |
| | Actividad en fallos | |
| | Aseguramiento de Calidad | |
| | Historial problemas conocidos | |
| | Disponibilidad de resultados de evaluación | |
| | Herramientas de desarrollo | |
| | Probabilidad de bifurcación | |
| | Relación entre *stakeholders* | |
| Comunidad | Comunidad de desarrolladores | |
| | Comunidad de usuarios | |
| | Contribución al producto | |
| Eficiencia | Seguridad | |
| | Fiabilidad | |
| | Rendimiento | |
| | Escalabilidad | |
| | Estabilidad | |
| Entrenamiento (Capacitación / Documentación) | Documentación | |
| | Capacitación | |
| | Asesoramiento | |
| | Libros | |
| Integración | Modularidad | |
| | Colaboración con otros productos | |
| Portabilidad | Independencia de plataforma | |
| | Independencia de proveedor | |
| Software / Producto | Soporte | |
| | Edad | |
| | Licencias | |
| | Jerarquías humanas | |

Tabla 28. (Continuación)

| Criterio | Característica | Seleccionada |
|---|---|---|
| | Puntos de venta | |
| | Funcionalidades | |
| | Normas | |
| | Distribución | |
| | Informes | |
| | Tecnología probada | |
| Especificidad | Característica 1 | |
| | Característica 2 | |

Fuente: Modelo de evaluación para la selección de herramientas de software libre en el proceso de gestión documental. (Jiménez, 2016)

Para construir el artefacto de operacionalización (Ver Tabla 29), se deben seleccionar las características indicadas en la tabla 28, asociadas a sus respectivos criterios, también los objetivos que se desean alcanzar por cada una de ellas, posteriormente los indicadores que serán los elementos a evaluar directamente, por lo que se debe formular al menos una pregunta por cada uno de ellos para medir su nivel de cumplimiento, el modelo recomienda realizar la evaluación utilizando la métrica definida en la tabla 27, la cual da una valoración de uno (1) a cinco (5).

Tabla 29. Artefacto de operacionalización.

| | Objetivo | Característica | Indicador | Pregunta | Métrica |
|---|---|---|---|---|---|
| Criterio | Objetivo 1. | Característica 1. | Indicador 1. | Pregunta 1 | Métrica 1. |
| | | | | Pregunta 2 | Métrica 2. |
| | | | | …. | …. |
| | | | | Pregunta N | Métrica N |
| | | | Indicador 2. | …. | …. |
| | | | …. | …. | …. |
| | | | Indicador M. | | |
| | Objetivo 2. | Característica 2. | …. | …. | …. |
| | | …. | …. | …. | …. |
| | Objetivo L | Característica L | …. | …. | …. |

Fuente: Modelo de evaluación para la selección de herramientas de software libre en el proceso de gestión documental. (Jiménez, 2016)

Finalmente dentro de la fase de planeación se debe construir el cuestionario de evaluación, en donde se deben trasladar los indicadores, las preguntas y las métricas desarrolladas en la matriz de operacionalización GQM del modelo a un cuestionario como se muestra en la figura 15.

Figura 15. Instrumento de Evaluación.

Fecha: _____
Evaluador: _____
Herramienta: _____

5. Totalmente de acuerdo, 4. De acuerdo, 3. Ni de acuerdo, ni en desacuerdo, 2. En desacuerdo, 1. Totalmente en desacuerdo

| | Caracteristica | Indicador | Preguntas | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| Criterio 1 | Caracteristica 1 | Indicador 1 | ¿Pregunta 1? | | | | | |
| | | Indicador 2 | ¿Pregunta 2? | | | | | |
| | | Indicador 3 | ¿Pregunta 3? | | | | | |
| | | ... | ... | | | | | |
| | | ... | ... | | | | | |
| | Caracteristica 2 | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | ... | ... | ... | | | | | |
| | Caracteristica n | Indicador m | ¿Pregunta y? | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

Fuente: Modelo de evaluación para la selección de herramientas de software libre en el proceso de gestión documental (Jiménez, 2016).

**Fase 2 - Ejecución:** En esta fase se procede a realizar la valoración de cada uno de los productos software teniendo en cuenta cada uno de los indicadores seleccionados en la etapa de planeación y el instrumento elaborado (Cuestionario).

**Fase 3 - Verificación:** se realiza una interpretación de los datos obtenidos en la etapa anterior y se los traslada a la tabla 30, registrando los valores obtenidos por cada indicador evaluado en cada una de las herramientas, finalmente se los totaliza por cada criterio obteniendo el valor total (TCn) de cada herramienta.

Tabla 30. Valoraciones por indicador de las herramientas evaluadas

| | | | **Herramientas Evaluadas** | | | |
|---|---|---|---|---|---|---|
| **Criterio** | **Característica** | **Indicador** | **H1** | **H2** | **…** | **Hn** |
| Criterio 1 | Característica 1 | Indicador 1 | Valor | | | |
| | | Indicador 2 | | | | |
| | | Indicador 3 | | | | |
| | | … | … | … | … | … |
| | | Indicador n | … | … | … | … |
| | Característica 2 | | | | | |
| | … | … | … | … | … | … |
| Total Criterio 1 | | | TC1 | TC2 | | Tcn |

Fuente: Modelo de evaluación para la selección de herramientas de software libre en el proceso de gestión documental. (Jiménez, 2016)

Una vez obtenido el valor total por cada criterio, se sintetiza estos resultados en una tabla de valoración final, como se observa en la tabla 31, en donde se debe multiplicar el total de cada criterio por la ponderación asignada en la fase de planeación, al final de la tabla se calcula el valor total obtenido de manera cuantitativa, haciendo la suma de los valores finales ponderados de cada uno de los criterios.

Tabla 31. Valoración Final

| Criterio | Ponderación | Herramientas Evaluadas | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | H1 | | H2 | | … | Hn | |
| | | TC | VF | TC | VF | | TC | VF |
| Criterio 1 | % | | | | | | | |
| Criterio 2 | % | | | | | | | |
| …. | … | | | | | | | |
| Criterio n | % | | | | | | | |
| **Total** | | | **##%** | | **##%** | | | **##%** |

Fuente: Modelo de evaluación para la selección de herramientas de software libre en el proceso de gestión documental. (Jiménez, 2016)

H: herramienta evaluada, TC: total criterio, VF: valor final multiplicación TC * ponderación

Posteriormente se trasladan los nombres de las herramientas evaluadas junto con sus porcentajes totales obtenidos a una tabla ordenados de manera descendente, tal como se indica en la tabla 32, en donde quedará en primer lugar la herramienta con el mayor porcentaje.

Tabla 32. Resultados Finales

| Herramienta evaluada | Valor Ponderado Final |
|---|---|
| Herramienta x | % |
| Herramienta y | % |
| ….. | … |

Fuente: Modelo de evaluación para la selección de herramientas de software libre en el proceso de gestión documental. (Jiménez, 2016)

**Fase 4 - Selección:** En esta fase el evaluador es quien toma la decisión de cuál de las herramientas va adoptar, con base en los resultados obtenidos, la herramienta selecciona se ubica en una tabla (Ver Tabla 33) junto con su valor ponderado final, el modelo propone realizar un proceso de argumentación en donde se justifique el por qué de su elección tomando como base los valores de la evaluación general de la herramienta.

Tabla 33. Selección Final Herramienta

| Herramienta seleccionada | Valor Ponderado Final |
|---|---|
| Herramienta n | % |

Fuente: Modelo de evaluación para la selección de herramientas de software libre en el proceso de gestión documental. (Jiménez, 2016)

Como se indicó en el Marco Teórico existen varias herramientas para la gestión del proceso Scrum, de las cuales por decisión del investigador se evaluarán las siguientes Kunagi, Scrumpy, Sprintometer, IceScrum y GanttProject. De acuerdo al modelo a utilizar se procedió a ejecutar cada una de sus fases:

**Fase 1 – Planeación:** En ésta fase se realizó la ponderación de cada uno de los nueve criterios, en donde el evaluador según su criterio asignó el nivel de importancia como se indica en la Tabla 34.

Tabla 34. Ponderación de criterios

| Criterio | Ponderación |
|---|---|
| Aceptación / Usabilidad | 20% |
| Administración | 10% |
| Comunidad | 0% |
| Eficiencia | 10% |
| Entrenamiento (Capacitación/Documentación) | 10% |
| Integración | 10% |
| Portabilidad | 10% |
| Software / Producto | 5% |
| Especificidad | 25% |
| **Total** | **100%** |

Fuente: La presente investigación 2017

Luego siguiendo el modelo se debe operacionalizar las características a evaluar, para lo cual se utilizará una escala para la evaluación como se indica en la tabla 27.

Tabla 35. Escala de Evaluación

| Valoración | Juicio |
|---|---|
| **5** | Muy Alto |
| **4** | Alto |
| **3** | Ni Alto, Ni bajo |
| **2** | Bajo |
| **1** | Muy Bajo |

Fuente: La presente investigación 2017

Aunque el modelo define las características a evaluar por cada criterio, por ser un modelo abierto el evaluador puede determinar cuáles selecciona o propone. En la tabla 36 se visualizan las características definidas para las herramientas a evaluar.

Tabla 36. Características a Evaluar

| Criterio | Característica | Seleccionada |
|---|---|---|
| Aceptación / Usabilidad | Usabilidad | **X** |
| | Penetración en el mercado | |
| | Interfaz de usuario | **X** |
| | Mantenibilidad | |

Tabla 36. (Continuación)

| Criterio | Característica | Seleccionada |
|---|---|---|
| | Implementación | X |
| | Facilidad de despliegue | |
| Administración | Pruebas | |
| | Actividad de versiones | X |
| | Actividad en características | |
| | Actividad en fallos | |
| | Aseguramiento de Calidad | |
| | Historial problemas conocidos | |
| | Disponibilidad de resultados de evaluación | |
| | Herramientas de desarrollo | |
| | Probabilidad de bifurcación | |
| | Relación entre Stakeholders | X |
| Comunidad | Comunidad de desarrolladores | |
| | Comunidad de usuarios | |
| | Contribución al producto | |
| Eficiencia | Seguridad | X |
| | Fiabilidad | X |
| | Rendimiento | X |
| | Escalabilidad | |
| | Estabilidad | X |
| Entrenamiento (Capacitación / Documentación) | Documentación | X |
| | Capacitación | X |
| | Asesoramiento | |
| | Libros | |
| Integración | Modularidad | |
| | Colaboración con otros productos | X |
| Portabilidad | Independencia de plataforma | X |
| | Independencia de proveedor | |
| Software / Producto | Soporte | X |
| | Edad | X |
| | Licencias | |
| | Jerarquías humanas | |
| | Puntos de venta | |
| | Funcionalidades | X |
| | Normas | |
| | Distribución | |
| | Informes | |
| | Tecnología probada | |

Fuente: La presente investigación 2017

El siguiente paso en la planeación, consiste en la operacionalización de los indicadores, por tanto, se procede a completar la información de medición de cada una de las características seleccionadas junto con los indicadores asociados, tal como se indica en la tabla 37. Para el criterio de Especificidad se han definido actividades y elementos propios de Scrum como los roles*, product backlog*, *sprint backlog*, *planning meeting*, *sprint review*, el *done* entre otros.

Tabla 37. Artefacto de Operacionalización

| Criterios | Objetivo | Característica | Indicador | Pregunta |
|---|---|---|---|---|
| **Aceptación / Usabilidad** | Comprobar el nivel de usabilidad de la herramienta | Usabilidad | Alertas | ¿Existen en el programa indicadores y alertas de progreso y error? |
| | | | Navegación | ¿Es posible trabajar en el aplicativo solo haciendo uso del teclado? |
| | | | Ayuda | ¿Cuál es el nivel de ayuda que posee el programa, dónde expliquen el funcionamiento de sus opciones? |
| | Determinar la fluidez y facilidad de uso de la interfaz de usuario | Interfaz de usuario | Simplicidad | ¿Qué tan intuitiva es la interfaz de usuario? |
| | | | Comprensibilidad | ¿Qué tan fácil son de relacionar los menús e iconos con las acciones a realizar? |
| | | | Internacionalización | ¿Qué tan fácil es de configurar el aplicativo en un idioma diferente? |
| | | | Configurabilidad | ¿Qué tan configurable es la apariencia del aplicativo? |
| | Comprobar la facilidad de implementación de la herramienta | Implementación | Conocimiento | ¿Qué tanto conocimiento técnico se necesita para su instalación? |
| | | | Tiempo | ¿Cuál es el porcentaje de tiempo para la instalación? |
| | | | Requisitos | ¿Cuál es el número necesario de otras herramientas para la instalación del software? |

Tabla 37. (Continuación)

| Criterios | Objetivo | Característica | Indicador | Pregunta |
|---|---|---|---|---|
| **Administración** | Identificar el nivel de actividad con base en el control de versiones del producto | Actividad de versiones | Tiempo | ¿Cuál es el nivel de control para las versiones del producto? |
| | Comprobar el nivel de relación e interacción de *stakeholders* | Relación entre *stakeholders* | Colaboración | ¿Cuál es el nivel de colaboración que permite la herramienta entre los integrantes del equipo? |
| | | | Interacción | ¿Cuál es el nivel de interacción que permite el aplicativo de los integrantes del equipo? |
| **Eficiencia** | Determinar el nivel de seguridad ofrecido por el producto | Seguridad | Usuarios | ¿Qué nivel de facilidad ofrece para la gestión de usuarios? |
| | | | Grupos | ¿Qué nivel de facilidad ofrece para la gestión de grupos? |
| | | | Permisos | ¿Qué nivel de facilidad ofrece para la gestión de permisos? |
| | Determinar la confiabilidad del producto | Fiabilidad | Integridad | ¿Qué nivel de recuperación de los datos tiene ante la presencia de fallas? |
| | Comprobar el rendimiento del producto | Rendimiento | Velocidad | ¿Cuál es el nivel de respuesta en la ejecución de las acciones críticas del software? |
| | Determinar el nivel de estabilidad que tiene el producto | Estabilidad | Estabilidad | ¿Qué tan estable es la herramienta al momento de trabajar? |
| **Entrenamiento (Capacitación/Documentación)** | Identificar la calidad de la documentación del producto a evaluar. | Documentación | Disponibilidad | ¿Qué nivel de documentación está disponible en la web? |
| | | | Actualización | ¿Cuál es el nivel de actualización de la documentación existente con relación a la última versión del software? |
| | Identificar el nivel de capacitación que debe tener una persona para utilizar el producto a evaluar | Capacitación | Tiempo | ¿Cuál es el porcentaje de tiempo que se requiere capacitar a una persona para usar el software? |

Tabla 37. (Continuación)

| Criterios | Objetivo | Característica | Indicador | Pregunta |
|---|---|---|---|---|
| **Integración** | Determinar el grado de colaboración del producto con otros aplicativos | Colaboración con otros productos | Colaboración | ¿Cuál es el nivel de integración que permite con otras aplicaciones? |
| **Portabilidad** | Identificar la independencia de sistema operativo | Independencia de plataforma | Multiplataforma | ¿En cuántas plataformas se puede instalar la herramienta? |
| | Identificar la independencia del ambiente de trabajo | | Ambiente | ¿En cuántos ambientes (Escritorio, WEB, Móvil, Servicio WEB y consola) se puede usar la herramienta? |
| **Software / Producto** | Comprobar el nivel de soporte ofrecido por el producto | Soporte | Fallas | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de fallas? |
| | | | Inquietudes | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de inquietudes? |
| | Identificar la madurez del producto | Edad | Madurez | ¿Cuál es el nivel de madurez del producto software? |
| | | | Actualidad | ¿Cuál es el nivel de frecuencia con el que se actualiza el software? |
| | Determinar las funcionalidades ofrecidas por el producto | Funcionalidad | Configurabilidad | ¿Qué nivel de parametrización tiene el software? |
| Especificidad | Determinar el aporte a la gestión de roles | Administración | Roles | ¿Qué nivel de facilidad ofrece para la gestión de roles? |
| | Determinar la gestión que el producto realiza a los proyectos de software bajo la metodología Scrum | Contenido | Requisitos | ¿Qué nivel de facilidad ofrece para la identificación de las necesidades del cliente para la construcción del product *backlog*? |
| | | | | ¿Con que facilidad se puede construir el sprint *backlog*? |
| | | | | ¿Con que facilidad se puede priorizar las necesidades del product *backlog*? |

Tabla 37. (Continuación)

| Criterios | Objetivo | Característica | Indicador | Pregunta |
|---|---|---|---|---|
| **Especificidad** | Determinar la gestión que el producto realiza a los proyectos de software bajo la metodología Scrum | Contenido | Requisitos | ¿Cuál es el nivel de facilidad que ofrece para la planeación del sprint *planning meeting*? |
| | | | | ¿Cuál es el nivel de facilidad que ofrece para el desarrollo del sprint *planning meeting*? |
| | | | | ¿Cuál es el nivel de facilidad que ofrece para la evaluación del sprint *planning meeting*? |
| | | | | ¿Qué nivel de facilidad ofrece para el desarrollo del sprint? |
| | | | | ¿Cuál es el nivel de facilidad que ofrece para la realización del sprint review? |
| | | | | ¿Con que facilidad se puede realizar la retrospectiva? |
| | | | | ¿Cuál es el nivel para el registro de métricas? |
| | | | | ¿Qué nivel de facilidad ofrece para la recopilación de datos relacionados con las métricas? |
| | | | | ¿Qué nivel la información ofrece a partir de los datos recopilados en las métricas? |
| | | | | ¿Con qué facilidad se pueden definir el Done? |
| | | | | ¿Con qué facilidad se puede gestionar la información de las métricas, a partir de los datos recopilados en los diferentes Sprints? |
| | | | | ¿Cuál es el nivel de facilidad con el que se puede gestionar visualmente las necesidades definidas en el product *backlog*? |
| | | | | ¿Qué nivel de facilidad ofrece para la gestión del daily *meeting*? |

Tabla 37. (Continuación)

| Criterios | Objetivo | Característica | Indicador | Pregunta |
|---|---|---|---|---|
| Especificidad | Determinar la gestión que el producto realiza a los proyectos de software bajo la metodología Scrum | Contenido | Requisitos | ¿Cuál es el nivel de facilidad que ofrece para asignar tiempo a las actividades del *done*? |
| | Determinar el nivel de aporte que se realiza al flujo de trabajo dentro del proyecto | Flujo de trabajo | Simple | ¿La herramienta permite la creación de tareas por roles? |
| | | | Colaborativo | ¿Se puede realizar actividades de trabajo colaborativo entre los roles? |
| | Determinar el nivel de aporte a las características complementarias que maneja la metodología Scrum (Nico-Nico, Dojo,Move programming y motivación) | Complementos | Motivación | ¿Qué nivel de facilidad ofrece para la motivación de los integrantes del Scrum team? |
| | | | | ¿Qué nivel de facilidad ofrece para identificar el estado de ánimo del Scrum team? |
| | | | Colaboración | ¿Qué nivel de facilidad ofrece para compartir conocimiento entre los integrantes del Scrum team? |
| | | | | ¿Qué nivel de facilidad ofrece para solucionar problemas complejos entre los integrantes del Scrum team? |

Fuente: La presente investigación 2017

Siguiendo con la fase de planeación se elaboró el cuestionario o instrumento para la evaluación de las herramientas indicadas anteriormente. (Ver Anexo B).

**Fase 2 – Ejecución.** En esta fase se realizó la puntuación de cada uno de los indicadores, incluidos en el cuestionario creado en la fase anterior, para realizar este proceso en primer lugar se preparó el entorno de pruebas tomando una maquina física como anfitriona que consta de 1Tb de disco duro, 16Gb de RAM, procesador Intel core i7 de quinta generación, 2Gb de tarjeta gráfica con sistema operativo Windows 10, posteriormente se procedió a crear máquinas virtuales con sistemas operativos diferentes cuyas características se indican en la Tabla 38.

Tabla 38**.** Características máquinas virtuales

| Sistema operativo | Disco duro (Gb) | RAM (Gb) | Tarjeta de video (Gb) |
|---|---|---|---|
| Windows 7 | 256 | 8 | 1 |
| Linux Mint | 256 | 8 | 1 |
| Mac Os Sierra | 256 | 8 | 1 |

Fuente: La presente investigación 2017

Para la realización de las pruebas se contó con la participación de estudiantes de último semestre del programa de Ingeniería de Sistemas de la Universidad Mariana, quienes para la valoración de cada uno de los indicadores aparte de las herramientas instaladas en el entorno de pruebas, tuvieron como fuentes de información las páginas oficiales y manuales de usuario de las mismas. Los datos obtenidos se pueden observar en el Anexo C.

**Fase 3 – Verificación.** En esta fase se interpretaran los resultados, en donde por cada criterio evaluado se presentará una tabla con el resumen por cada herramienta.

En la tabla 39, se presenta la síntesis para el criterio Aceptación/Usabilidad el cual tiene un total de 50 puntos para un máximo de 20%, en donde la herramienta GanttProject tuvo la más alta calificación con 35 puntos equivalente a un 14%, en segundo lugar la herramienta Kunagi con 34 puntos, estos son los lugares más relevantes y en el último lugar se encuentra la herramienta Scrumpy con 23 puntos y un porcentaje de 9.2%.

Tabla 39. Evaluación de Criterio Aceptación / Usabilidad

| | | | Herramientas Evaluadas | | | | |
|---|---|---|---|---|---|---|---|
| Criterio | Característica | Indicador | Kunagi | Scrumpy | Sprintometer | Icesrum | GanttProject |
| Aceptación/ Usabilidad | Usabilidad | Alertas | 3 | 4 | 4 | 3 | 2 |
| | | Navegación | 1 | 2 | 3 | 1 | 1 |
| | | Ayuda | 5 | 1 | 3 | 2 | 1 |
| | Interfaz de usuario | Simplicidad | 4 | 1 | 3 | 5 | 4 |
| | | Comprensibilidad | 5 | 3 | 3 | 5 | 5 |
| | | Internacionalización | 1 | 1 | 1 | 5 | 5 |
| | | Configurabilidad | 1 | 1 | 1 | 3 | 2 |
| | Implementación | Conocimiento | 5 | 2 | 5 | 1 | 5 |
| | | Tiempo | 5 | 3 | 5 | 1 | 5 |
| | | Requisitos | 4 | 5 | 5 | 4 | 5 |
| Total Aceptación/Usabilidad | | | 34 | 23 | 33 | 30 | 35 |
| % por criterio (50 puntos = 20%) | | | 13.6 | 9.2 | 13.2 | 12 | 14 |

Fuente: La presente investigación 2017

En la Tabla 40, se muestran los resultados para el criterio de Administración en los cuales la herramientas Kunagi y IceScrum ocupan el primer lugar con un total de 15 puntos y un porcentaje de 10%, en segundo lugar, se encuentra la herramienta Scrumpy con 7 puntos y un porcentaje de 4.7%, en tercer y último lugar se encuentran las herramientas Sprintometer y GanttProject con 3 puntos y 2%.

Tabla 40. Evaluación de Criterio Administración

| | | | Herramientas Evaluadas | | | | |
|---|---|---|---|---|---|---|---|
| Criterio | Característica | Indicador | Kunagi | Scrumpy | Sprintometer | Icesrum | GanttProject |
| Administración | Actividad de versiones | Tiempo | 5 | 5 | 1 | 5 | 1 |
| | Relación entre stakeholders | Colaboración | 5 | 1 | 1 | 5 | 1 |
| | | Interacción | 5 | 1 | 1 | 5 | 1 |
| Total Administración | | | 15 | 7 | 3 | 15 | 3 |
| % por criterio (15 puntos = 10%) | | | 10 | 4.7 | 2 | 10 | 2 |

Fuente: La presente investigación 2017

En la Tabla 41, se puede observar los resultados para el criterio de Eficiencia, el cual tiene una ponderación del 10% equivalente a un total de 30 puntos, en el que la herramienta IceScrum ocupa el primer lugar con un puntaje ideal de 30 puntos obteniendo el 10%, en segundo lugar, se encuentra Kunagi con 28 puntos y 9.3%, en tercer lugar se encuentra GanttProject con 18 puntos y un porcentaje de 6%, en

penúltimo lugar se encuentra Sprintometer con 15 puntos y un porcentaje de 5% y en el último lugar Scrumpy con una puntuación de 14 y un porcentaje de 4.7%.

Tabla 41. Evaluación de Criterio Eficiencia

| | | | Herramientas Evaluadas | | | | |
|---|---|---|---|---|---|---|---|
| **Criterio** | **Característica** | **Indicador** | **Kunagi** | **Scrumpy** | **Sprintometer** | **Icesrum** | **GanttProject** |
| **Eficiencia** | **Seguridad** | **Usuarios** | 5 | 1 | 1 | 5 | 2 |
| | | **Grupos** | 5 | 1 | 1 | 5 | 1 |
| | | **Permisos** | 5 | 1 | 1 | 5 | 1 |
| | **Fiabilidad** | **Integridad** | 5 | 1 | 4 | 5 | 4 |
| | **Rendimiento** | **Velocidad** | 4 | 5 | 5 | 5 | 5 |
| | **Estabilidad** | **Estabilidad** | 4 | 5 | 3 | 5 | 5 |
| **Total Eficiencia** | | | **28** | **14** | **15** | **30** | **18** |
| **% por criterio (30 puntos =10%)** | | | **9.3** | **4.7** | **5** | **10** | **6** |

Fuente: La presente investigación 2017

La Tabla 42, se indican los resultados de la evaluación del criterio Entrenamiento con una ponderación del 10% y una máxima puntuación de 15, en el que Kunagi está en primer lugar con 13 puntos correspondientes al 8.7%, en segundo lugar IceScrum con 12 puntos, en tercer lugar Scrumpy con 10 puntos y un 6.7% y en último lugar Sprintometer y GanttProject con 8 puntos equivalentes a un 5.3%.

Tabla 42. Evaluación de Criterio Entrenamiento

| | | | Herramientas Evaluadas | | | | |
|---|---|---|---|---|---|---|---|
| **Criterio** | **Característica** | **Indicador** | **Kunagi** | **Scrumpy** | **Sprintometer** | **Icesrum** | **GanttProject** |
| **Entrenamiento** | **Documentación** | **Disponibilidad** | 5 | 5 | 5 | 5 | 2 |
| | | **Actualización** | 5 | 2 | 2 | 3 | 2 |
| | **Capacitación** | **Tiempo** | 3 | 3 | 1 | 4 | 4 |
| **Total Entrenamiento** | | | **13** | **10** | **8** | **12** | **8** |
| **% por criterio (15 puntos = 10%)** | | | **8.7** | **6.7** | **5.3** | **8** | **5.3** |

Fuente: La presente investigación 2017

En la Tabla 43, se puede observar el resumen de valoración para el criterio Integridad, en donde se tiene como resultado la puntuación más baja de 1 punto y el 2% para todas las herramientas.

Tabla 43. Evaluación de Criterio Integración

| | | | Herramientas Evaluadas | | | | |
|---|---|---|---|---|---|---|---|
| **Criterio** | **Característica** | **Indicador** | **Kunagi** | **Scrumpy** | **Sprintometer** | **Icesrum** | **GanttProject** |
| **Integración** | **Colaboración con otros productos** | **Colaboración** | 1 | 1 | 1 | 1 | 1 |
| **Total Integración** | | | **1** | **1** | **1** | **1** | **1** |
| **% por criterio (5 puntos = 10%)** | | | **2** | **2** | **2** | **2** | **2** |

Fuente: La presente investigación 2017

Para el criterio de portabilidad como se puede observar en Tabla 44, el primer lugar lo obtuvo la herramienta Kunagi con una calificación de 7 puntos, equivalente a un 7%, en la siguiente posición están las herramientas IceScrum, Scrumpy, GanttProject con un total de 6 puntos correspondiente al 6%, y en el último lugar Sprintometer con 2 puntos y un 2%.

Tabla 44. Evaluación de Criterio Portabilidad

| | | | Herramientas Evaluadas | | | | |
|---|---|---|---|---|---|---|---|
| **Criterio** | **Característica** | **Indicador** | **Kunagi** | **Scrumpy** | **Sprintometer** | **Icesrum** | **GanttProject** |
| **Portabilidad** | **Independencia de plataforma** | **Multiplaforma** | 5 | 5 | 1 | 5 | 5 |
| | | **Ambiente** | 2 | 1 | 1 | 1 | 1 |
| **Total Portabilidad** | | | **7** | **6** | **2** | **6** | **6** |
| **% por criterio (10 puntos = 10%)** | | | **7** | **6** | **2** | **6** | **6** |

Fuente: La presente investigación 2017

En la Tabla 45, se indican los resultados para el criterio Software/Producto en el que la herramienta con mayor calificación fue IceScrum con 21 puntos de 25 posibles, equivalente a un 4.2%, en segundo lugar se encuentra Kunagi con 17 puntos y un 3.4%, en tercer lugar Scrumpy con 12 puntos y 2.4% y en el último lugar GanttProject con 10 puntos y 2%.

Tabla 45. Evaluación de Criterio Software/Producto

| | | | Herramientas Evaluadas | | | | |
|---|---|---|---|---|---|---|---|
| Criterio | Característica | Indicador | Kunagi | Scrumpy | Sprintometer | Icesrum | GanttProject |
| Software/ Producto | Soporte | Fallas | 2 | 2 | 2 | 3 | 3 |
| | | Inquietudes | 2 | 2 | 2 | 3 | 3 |
| | Edad | Madurez | 5 | 5 | 5 | 5 | 1 |
| | | Actualidad | 3 | 1 | 3 | 5 | 1 |
| | Funcionalidad | Configurabilidad | 5 | 2 | 4 | 5 | 2 |
| Total Software/Producto | | | 17 | 12 | 16 | 21 | 10 |
| % por criterio (25 puntos = 5%) | | | 3.4 | 2.4 | 3.2 | 4.2 | 2 |

Fuente: La presente investigación 2017

En la Tabla 46, se indica la valoración dada al criterio especificidad, el cual tiene una ponderación del 25% y 120 puntos posibles, en donde las herramientas que tuvieron mejor desempeño en la gestión del proceso Scrum fueron en primer lugar Kunagi con 91 puntos equivalentes al 19%, en segundo lugar IceScrum con 72 puntos y 15%, en tercer lugar Sprintometer con 48 puntos y 10%, en cuarto lugar GanttProject con 42 puntos y un 8.8% y en el último lugar Scrumpy con 35 puntos y 7.3%.

Tabla 46. Evaluación de Criterio Especificidad

| | | | Herramientas Evaluadas | | | | |
|---|---|---|---|---|---|---|---|
| Criterio | Característica | Indicador | Kunagi | Scrumpy | Sprintometer | Icesrum | GanttProject |
| Especificidad | Administración | Roles | 5 | 1 | 1 | 5 | 3 |
| | Contenido | Requisitos | 1 | 1 | 1 | 4 | 1 |
| | | | 4 | 5 | 4 | 5 | 1 |
| | | | 5 | 2 | 2 | 4 | 3 |
| | | | 5 | 1 | 1 | 1 | 1 |
| | | | 1 | 1 | 1 | 1 | 1 |
| | | | 4 | 1 | 1 | 1 | 1 |
| | | | 5 | 2 | 4 | 5 | 1 |
| | | | 4 | 1 | 1 | 1 | 1 |
| | | | 4 | 1 | 1 | 3 | 1 |
| | | | 1 | 1 | 1 | 1 | 1 |
| | | | 4 | 2 | 4 | 4 | 2 |
| | | | 3 | 2 | 5 | 5 | 2 |
| | | | 5 | 1 | 4 | 4 | 4 |
| | | | 1 | 1 | 2 | 4 | 1 |
| | | | 5 | 4 | 4 | 5 | 5 |
| | | | 4 | 1 | 1 | 1 | 1 |
| | | | 5 | 1 | 4 | 5 | 4 |
| | Flujo de trabajo | Simple | 5 | 1 | 1 | 5 | 3 |
| | | Colaborativo | 5 | 1 | 1 | 4 | 1 |
| | Complementos | Motivación | 4 | 1 | 1 | 1 | 1 |
| | | | 3 | 1 | 1 | 1 | 1 |
| | | Colaboración | 4 | 1 | 1 | 1 | 1 |
| | | | 4 | 1 | 1 | 1 | 1 |
| Total Especificidad | | | 91 | 35 | 48 | 72 | 42 |
| % por criterio (120 puntos = 25%) | | | 19 | 7.3 | 10 | 15 | 8.8 |

Fuente: La presente investigación 2017

Finalmente se realiza el consolidado de las puntuaciones obtenidas por cada herramienta en cada uno de los criterios seleccionados como lo indica la columna TC, también en la columna VF se muestra el valor total de cada criterio ponderado de acuerdo al porcentaje asignado por el evaluador en cada uno de los criterios en la etapa de planeación, obteniendo así un valor total ponderado resultado de la sumatoria de los valores por cada herramienta.

Tabla 47. Valoración Final

| Criterio | Ponderación | Herramientas Evaluadas | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Kunagi | | Scrumpy | | Sprintometer | | Icescrum | | GanttProject | |
| | | TC | VF | TC | VF | TC | VF | TC | VF | TC | VF |
| **Aceptación/Usabilidad** | **20%** | 34 | 13.6 | 23 | 9.2 | 33 | 13.2 | 30 | 12 | 35 | 14 |
| **Administración** | **10%** | 15 | 10 | 7 | 4.7 | 3 | 2 | 15 | 10 | 3 | 2 |
| **Eficiencia** | **10%** | 28 | 9.3 | 14 | 4.7 | 15 | 5 | 30 | 10 | 18 | 6 |
| **Entrenamiento** | **10%** | 13 | 8.7 | 10 | 6.7 | 8 | 5.3 | 12 | 8 | 8 | 5.3 |
| **Integración** | **10%** | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| **Portabilidad** | **10%** | 7 | 7 | 6 | 6 | 2 | 2 | 6 | 6 | 6 | 6 |
| **Software/Producto** | **5%** | 17 | 3.4 | 12 | 2.4 | 16 | 3.2 | 21 | 4.2 | 10 | 2 |
| **Especificidad** | **25%** | 91 | 19 | 35 | 7.3 | 48 | 10 | 72 | 15 | 42 | 8.8 |
| | **Total (%)** | | **73** | | **42.9** | | **42.7** | | **67.2** | | **46.1** |

Fuente: La presente investigación 2017

Finalmente como indica el modelo utilizado se debe trasladar los nombres de las herramientas evaluadas junto con los porcentajes obtenidos, a una tabla ordenando los resultados de manera descendente, tal como se muestra en la tabla 48.

Tabla 48. Resultados Finales de Evaluación

| Herramienta Evaluada | Valor Ponderado Final |
|---|---|
| Kunagi | 73% |
| IceScrum | 67.2% |
| GanttProject | 46.1% |
| Scrumpy | 42.9% |
| Sprintometer | 42.7% |

Fuente: La presente investigación 2017

**Fase 4 – Selección**

Por último siguiendo el modelo se aplicó la fase de selección, en donde la herramienta Kunagi es la alternativa más recomendable, ya que obtuvo un ponderado final de 73%.

Tabla 49. Selección de la herramienta por el evaluador

| Herramienta seleccionada | Valor Ponderado Final |
|---|---|
| Kunagi | 73% |

Fuente: La presente investigación 2017

A continuación se presenta las ventajas y desventajas encontradas para esta herramienta en la fase de evaluación, para las demás herramientas observar el Anexo D.

Para el criterio de Aceptación/Usabilidad Kunagi es una herramienta muy intuitiva, que permite relacionar fácilmente cada uno de sus menús e iconos con la función a realizar, también posee texto y pop ups de ayuda en cada área de trabajo indicando de que trata un elemento y que se debería registrar en él, en caso de posibles problemas se indican mediante mensajes de error y alertas al usuario. Algunas desventajas están relacionadas con el idioma que solo viene en Ingles y la mayoría de las funciones se debe realizar con el mouse y en cuanto a la apariencia no es posible cambiarla, ya tiene unos estilos y colores por defecto.

En el criterio Administración cuenta con buena documentación en sus diferentes versiones, en lo relacionado a la interacción de los *stakeholders* la herramienta cuenta con wikis, blogs, fotos que permiten la colaboración y visualización del trabajo simultáneamente.

Dentro del criterio Eficiencia en cuanto a la seguridad posee autenticación a través de usuario y contraseña, cuando se registra un nuevo usuario es posible asignarle un rol, el cual cuenta con ciertos privilegios de acuerdo a las directrices de la metodología Scrum, permitiendo el acceso únicamente a las actividades o

elementos indicados para este usuario. Además cuenta con su propia base de datos en la cual almacena la información sobre los proyectos y sus especificidades, realizando backups de la misma varias veces al día, lo cual permite restaurar los datos ante una posible falla o error.

Evaluando el criterio Entrenamiento en lo relacionado a documentación Kunagi cuenta con un buen nivel de información actualizada en su página web principalmente a través de guías y blogs, que van desde indicar el proceso de descarga e instalación, hasta la solución de errores detectados por los usuarios en cada una de las versiones. Una vez se ha instalado la herramienta su manejo es muy fácil e intuitivo lo que disminuye bastante la curva de aprendizaje de un nuevo usuario, claro está siempre y cuando se conozca los elementos y conceptos de la metodología Scrum.

En el criterio Integración si hay una desventaja notable, ya que Kunagi no permite integrarse con ninguna herramienta de desarrollo. Aunque esto se compensa de cierta manera al evaluar el criterio de Portabilidad donde la herramienta está disponible para todos los sistemas operativos ya sea como aplicación de escritorio o Web.

Para el criterio Software/Producto en cuanto al soporte Kunagi cuenta en su página con un formulario donde se pueden realizar las consultas y también visualizar el historial de las mismas realizadas por otros usuarios, adicionalmente están clasificadas por temas para facilitar su navegación, A pesar de que es una herramienta que cuenta con un grado de madurez de alrededor de 6 años, una desventaja es que las actualizaciones son casi anualmente, dificultando la evolución del producto.

Y para finalizar, al evaluar el criterio Especificidad en donde se califica la gestión de los diferentes elementos de la metodología Scrum (Etapa, Actividad, Rol, Artefactos y Lineamiento), se comprobó que Kunagi incorpora muy bien cada uno de ellos con sus diferentes componentes por ejemplo permite administrar los diferentes roles y asigna automáticamente los privilegios conforme lo indica Scrum, también crear el Producto *Backlog* y el *Sprint Backlog* es muy fácil los atributos solicitados para las historias de usuario son fáciles de identificar y diligenciar, es posible realizar la

estimación de las historias utilizando la técnica de Planning Poker. A demás el uso de menús contextuales facilita la identificación de las actividades a realizar sobre el elemento seleccionado.

Algo muy interesante de Kunagi es la definición del Done, en donde por cada Historia de Usuario se pueden definir qué actividades se deben realizar y cuánto tiempo tomará hacerlas, visualizándolas en un tablero Kanban que consta de tres columnas, las tareas por hacer, las que se están realizando y las terminadas dando al usuario un control del proceso de desarrollo del *Sprint*. Una vez se finaliza es posible registrar las conclusiones de la reunión del *Sprint Review* y el *Sprint Retrospective* o retrospectiva.

Algo importante es que la herramienta permite al líder del proyecto gestionar fácilmente la asignación de las diferentes actividades a cada integrante y seleccionar el estado de ánimo con el cual se desarrolla la actividad o con la cual se la termina. También permite la creación de blogs, wikis y chats para el equipo con el fin de compartir conocimiento o para resolver problemas presentados dentro del desarrollo de alguna actividad.

Para el elemento lineamiento Kunagi permite a través del grafico conocido como el *Burndown Chart* visualizar la complejidad que está manejando el grupo y poder determinar el tiempo de trabajo que falta para completar el *Sprint* y de esta forma saber si es necesario realizar ajustes para cumplir con el Incremento de dicho *Sprint*.

Entre otras de las herramientas con las que cuenta la aplicación está el calendario, en el cual se pueden programar reuniones, actividades etc. También es posible subir archivos importantes a un repositorio del proyecto.

Aunque como se indicó anteriormente las ventajas que posee esta herramienta son muchas, de la misma manera hay algunas desventajas como la falta de un sitio en el cual se pueda desarrollar y evaluar el *planning meeting* y registrar métricas a evaluar diferentes a las que trae por defecto, además haría falta generar un reporte comparativo entre Sprints.

Una vez dados los argumentos por los cuales la herramienta Kunagi fue la que obtuvo mayor puntuación en la evaluación, para el desarrollo del siguiente objetivo que consiste en Aplicar la propuesta de trabajo basada en Scrum y apoyada por una herramienta de Software libre, se recomendará al Centro de Informática trabajar con Kunagi para contribuir en el proceso de creación de un producto software utilizando Scrum.

## 2.4 FORMULACIÓN DE LA PROPUESTA DE TRABAJO BASADA EN SCRUM, APOYADA POR HERRAMIENTAS DE SOFTWARE LIBRE PARA EL PROCESO DE CONSTRUCCIÓN DE SOFTWARE DEL CENTRO DE INFORMÁTICA DE LA UNIVERSIDAD DE NARIÑO

Teniendo en cuenta el trabajo realizado por Hernandez, Martínez, Argote, y Coral, (2015) donde plantean una forma de trabajo alternativa basada en Scrum, se ha tomado algunos lineamientos de la propuesta que se sintetizan en la Figura 16.

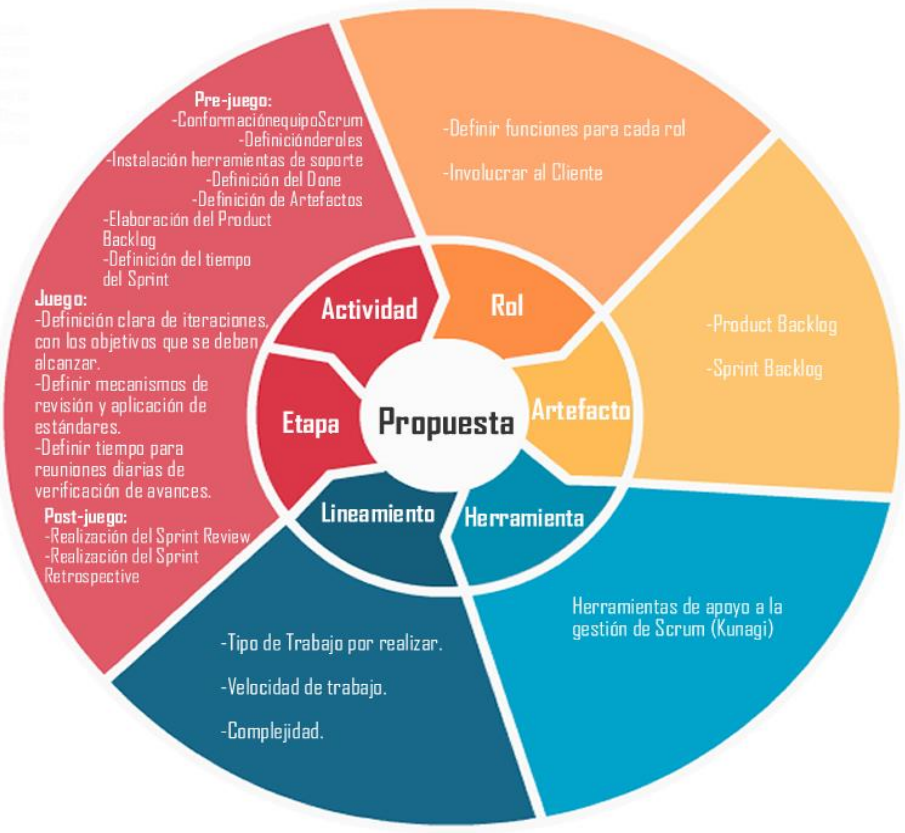
Con base en la propuesta de Hernandez, y Otros (2015), se plantea un camino para adoptar los lineamientos incluyendo técnicas que abordan problemas como: motivación, interrupciones del trabajo, socialización del conocimiento entre los integrantes del equipo y solución de problemas de manera colectiva; identificados en la caracterización de los elementos metodológicos del centro de informática en la construcción, soporte y mantenimiento de software, donde predomina el trabajo individual.

La propuesta plantea incorporar las técnicas “Niko Niko”, “Pomodoro”, “Mob Programming” y “Coding Dojo”. A continuación, se presenta una forma escalonada de adoptar la propuesta, en donde en cada nivel se describe la razón por la cual se incluye cada técnica, como respuesta a cada una de las necesidades planteadas por los expertos al momento de crear software.

La técnica “Niko Niko[1]” también conocida como calendario de la felicidad o índice de la felicidad, consiste en organizar un calendario en el cual cada fila corresponde

[1] Para más información acerca de la técnica visitar http://www.geocities.jp/nikonikocalendar/index_en.html

Figura 16. Propuesta Adaptativa basada en Scrum




Fuente: Adaptado de Metodología adaptativa basada en Scrum: Caso empresas de la Industria de Software en San Juan de Pasto - Colombia (Hernandez, y Otros, 2015).

a un miembro del equipo y las columnas a los días, en donde cada integrante registrará su estado de ánimo antes de iniciar con la jornada de trabajo por medio de emoticones o caras que identifican a una persona en sus diferentes estados como: feliz, indiferente, triste, etc. Después de ciertos periodos de tiempo por ejemplo al realizar la retrospectiva del sprint o antes se revisa el calendario y se analizan los datos con el objetivo de detectar posibles problemas que puedan ocurrir y afecten la productividad de alguno de los integrantes, según (Leber, 2014) las personas más felices son aproximadamente 12% más productivas, de ahí que la técnica Niko Niko permitirá detectar amenazas a la productividad del *Development Team*, para aplicar correctivos a tiempo y mitigarlas.

La técnica “Pomodoro[2]” es un método utilizado para mejorar la administración del tiempo al realizar una tarea o actividad, en la cual se usa un reloj para dividir el tiempo en intervalos de 25 minutos llamados “Pomodoros”, en los que se trabaja sin interrupciones ni distracciones, estos se separan por pausas cortas de 5 a 10 minutos, que contribuyen a disminuir la fatiga mental y a que el trabajo se realice en menos tiempo.

La técnica “Mob Programming[3]”, es una estrategia de desarrollo de software en la que un grupo de desarrolladores (*Development Team*) trabajan sobre una tarea o actividad concreta y compleja al mismo tiempo en un mismo ordenador, en donde uno de los participantes “piloto” es el encargado de codificar, mientras que el resto “navegantes” opina y aporta ideas para solucionar el problema planteado, es importante que el piloto y los navegantes de vez en cuando cambien de funciones, de esta manera se fomenta el trabajo colaborativo y se mantiene al grupo activo al momento de retomar la tarea.

La técnica de “Coding Dojo[4]” se trata de una reunión de desarrolladores (*Development Team*), en la que se trabaja en un desafío de programación, en cuyo proceso cada participante hace sus aportes de acuerdo con sus conocimientos indiferentemente del nivel que tenga, lo que permite al grupo adquirir nuevas habilidades en técnicas y tecnologías de manera colaborativa y no competitiva.

Un aspecto importante a tener en cuenta al momento de adoptar la propuesta en el Centro de Informática es la dificultad que generaría el cambio en la forma como vienen trabajando los funcionarios, en donde deberán salir de su zona de confort y adoptar esta nueva propuesta. En ese sentido, se identificó un conjunto de adopciones incrementales para incorporar los elementos que plantea la propuesta tratando de que la transición o el cambio sea lo menos critico posible.

[2] Para más información acerca de la técnica visitar http://pomodorotechnique.com/

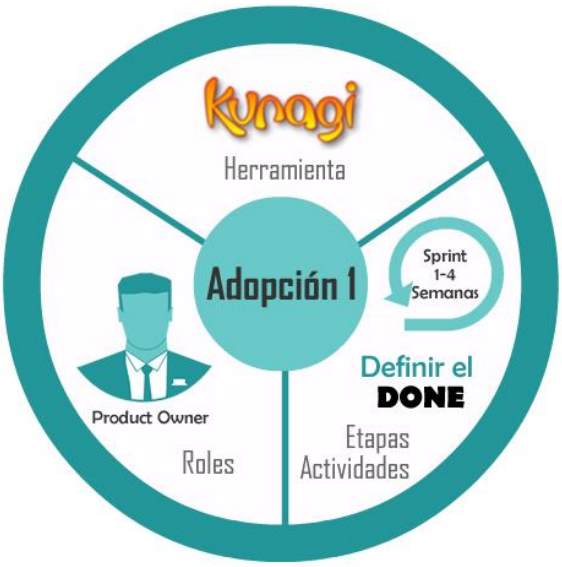
[3] Para más información acerca de la técnica Mob Programming visitar http://www.javiergarzas.com/2014/11/mob-programming.html

[4] Para más información acerca de la técnica Coding Dojo visitar https://www.genbetadev.com/metodologias-de-programacion/que-es-un-coding-dojo

En una primera etapa, como se muestra en la Figura 17, es muy importante establecer el rol del *Product Owner*, cuya función será la de recopilar las necesidades o requerimientos del cliente y traducirlos a historias de usuario. Para facilitar este proceso se instalará y configurará Kunagi[5], que fue la herramienta con mejores resultados aplicando el modelo OAM-F/OSS de evaluación de software libre. Esta herramienta permite gestionar el artefacto *Product Backlog*, cuya elaboración debe ser previa al inicio de la construcción, soporte y mantenimiento de un producto software, además debe realizarse de manera iterativa e incremental y compartirse con los demás miembros del equipo Scrum, para lo cual Kunagi cuenta con diferentes interfaces.

También es necesario definir un espacio de tiempo, que en Scrum se denomina como sprint. En este primer sprint, es recomendable definir las tareas del Done (hecho) que permitirán tener claro que se debe realizar para dar por finalizado un requerimiento o historia de usuario y puesto en funcionamiento en el cliente. Las tareas seleccionadas serán estrictamente aquellas que agreguen valor al producto software y su definición estará a cargo del *Development Team*.

Figura 17. Adopción primera de la propuesta

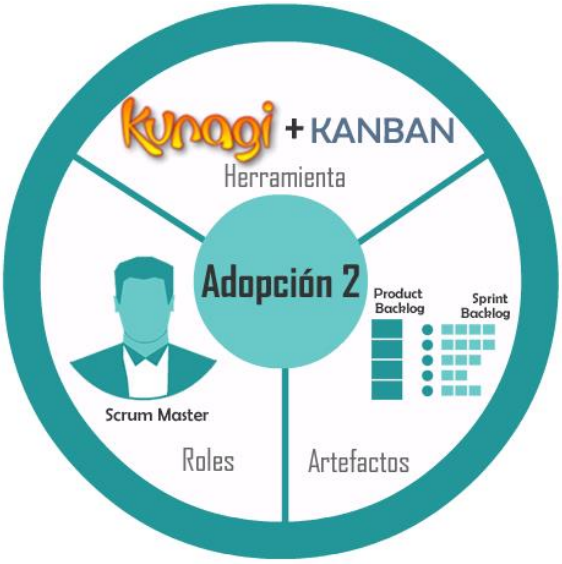



Fuente: La presente investigación 2017

[5] Para más información acerca de la herramienta visitar http://kunagi.org/

En una segunda etapa, como se muestra en la Figura 18, es necesario adicionar el rol del *Scrum Master,* el cual tendrá como funciones, contribuir en la adopción de Scrum aplicándose de forma correcta de acuerdo a la teoría, reglas y prácticas, asegurando de que sean entendidas por el equipo, por consiguiente también será un líder respaldando la autogestión del equipo y sus integrantes en la resolución de dificultades en el *Sprint* o en la dinámica de grupo. Para facilitar la realización de estas actividades la herramienta Kunagi permite por cada historia de usuario definir el *Done,* incluyendo las tareas a realizar con la asignación de sus responsables, en donde todos los integrantes del *Development Team* podrán observar el estado de las mismas de manera gráfica a través del tablero kanban [6] que provee Kunagi, lo cual les permitirá no iniciar tareas, sin haber finalizado de manera completa las que ya se habían priorizado. Como resultado de esta etapa y la anterior se habrá logrado la definición del Done y la elaboración de artefactos como el *Product Backlog*, del cual se seleccionará las historias de usuario con mayor prioridad para el sprint, generando el *Sprint Backlog*.

Figura 18. Adopción segunda de la propuesta




Fuente: La presente investigación 2017

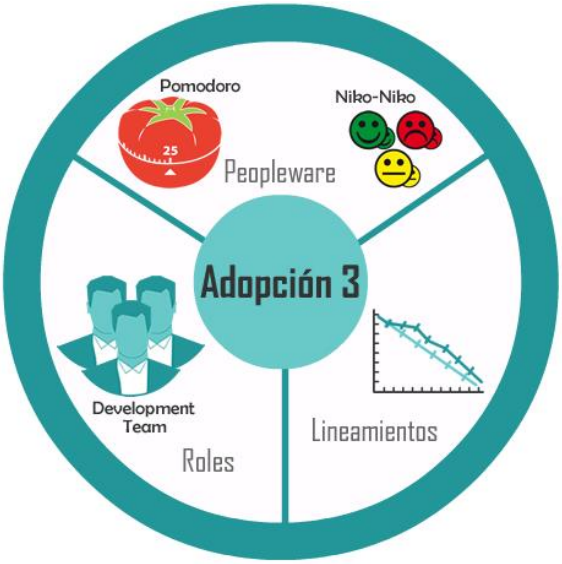
[6] Para más información sobre el tablero kanban visitar https://es.atlassian.com/agile/kanban

Para la tercera etapa, como se muestra en la Figura 19, el objetivo es empoderar (fortalecer) al *Development Team*, quien finalmente será el responsable de llevar a la práctica y realizar las tareas del Done, de ahí que deba estar en capacidad de autogestionarse y ser un equipo multifuncional, en el que por supuesto hay integrantes con especialidades concretas, pero donde el trabajo es solidario y la responsabilidad compartida con un propósito común que es lograr el incremento de cada *Sprint* y conseguir el mayor valor posible para la visión del cliente. En este sentido se utilizará la técnica "Niko Niko", la cual permite registrar el estado de ánimo de cada integrante, antes de iniciar su jornada laboral; esta información será muy valiosa para el Scrum Master al momento de buscar un equipo motivado, ya que será posible desarrollar acciones de intervención con el *Development Team.* Para las sesiones de trabajo, se propone utilizar la técnica "Pomodoro", con el propósito de mejorar la administración del tiempo en la realización de las tareas, buscando periodos de concentración de 25 minutos, con pausas entre ellos de 5 a 10 minutos, los expertos recomiendan hacerlo de manera iterativa hasta completar 4 o 5 periodos, de ahí en adelante las pausas pueden ser más prolongadas. El objetivo de utilizar esta técnica de manera iterativa en el desarrollo del trabajo del *Development Team* es fomentar una buena práctica, que a su vez se convierta en una métrica de productividad, que permita al grupo ser consiente de como organiza y ejecuta su trabajo en dichos periodos de tiempo.

En esta adopción una tarea muy importante que debe realizar el Scrum Master, es promover la gestión disciplinada (organizada y ordenada) de las tareas previamente definidas para el Done, permitiendo su visualización en la herramienta Kunagi, a través del *Burndown Chart* que es un reporte gráfico que muestra el trabajo que falta por realizarse para lograr el objetivo definido para el *Sprint* y así determinar cuál es el estado del proyecto (lineamiento).

Para la cuarta etapa, como se muestra en la Figura 20, se sugiere continuar con el empoderamiento del *Development Team*, aportando elementos para que los miembros de este equipo tengan la independencia de organizar y gestionar su trabajo, respetando las opiniones y aportes de todos incrementando de esta manera la eficiencia y efectividad de su trabajo. Para ello se utilizará la técnica "Mob programming", en la cual el equipo trabaja sobre una tarea o actividad concreta y compleja al mismo tiempo, en un mismo ordenador.

Figura 19. Adopción tercera de la propuesta



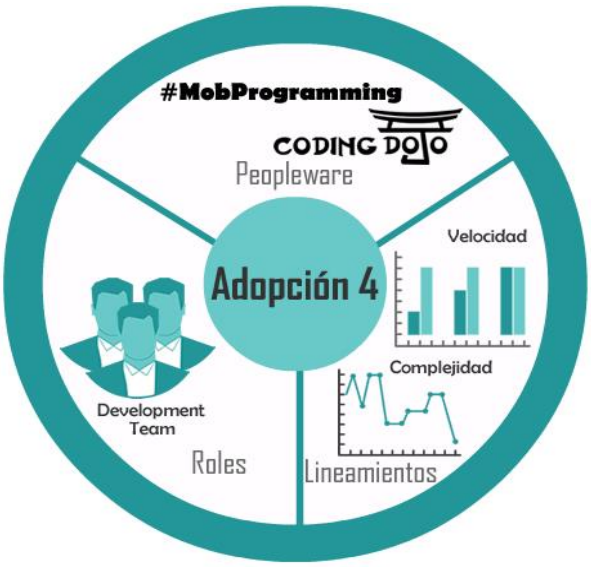

Fuente: La presente investigación 2017

Para que la experiencia y el conocimiento adquirido individualmente por los integrantes del equipo pueda ser trasmitido a los demás, se propone incorporar la técnica "*Coding Dojo*", para lo cual se programarán espacios de trabajo a todo el *Development Team* en los cuales se desarrollarán tareas propias del proyecto o aspecto nuevos por aprender y que serán de utilidad más adelante, fomentando de esta manera la construcción colectiva del conocimiento. En esta etapa al igual que en las anteriores el Scrum Master debe estar muy pendiente de que el equipo este coordinado y gestionando disciplinadamente las tareas del Done en la herramienta Kunagi, para que se puedan observar los diferentes reportes gráficos de la complejidad trabajada, dependiendo de la valoración dada a cada historia de usuario (lineamiento).

## 2.5 APLICACIÓN DE LA PROPUESTA

En esta sección se describe el proceso realizado para la aplicación de la propuesta de trabajo en equipo, basada en Scrum y *Peopleware*, soportada por la herramienta de software libre Kunagi en el Centro de Informática de la Universidad de Nariño, a través del desarrollo de un producto Software.

Figura 20. Adopción cuarta de la propuesta




Fuente: La presente investigación 2017

Para ello el Director de esta dependencia determinó aplicarla en el desarrollo del Proyecto "Administración Docentes-Carga Académica Liceo Universidad de Nariño", en donde la única solicitud realizada, fue que el tamaño del equipo no sea inferior a cuatro personas, ni mayor a nueve, ya que (Putnam, 2012) demuestra que en equipos de trabajo con más de siete (7) integrantes el esfuerzo se aumenta, pero el tiempo de trabajo del proyecto, no se acorta. Así mismo (Sutherland & Sutherland, 2014), plantean que el tamaño ideal de un equipo es de siete (7) integrantes y mínimo dos (2). En este sentido se seleccionó un equipo de trabajo de cuatro (4) integrantes, todos ingenieros de sistemas.

Para continuar con la aplicación de la propuesta era muy importante instalar y configurar la herramienta Kunagi, para esta tarea se asignó un servidor con sistema operativo Debian, en donde se realizó las configuraciones necesarias, permitiendo que la herramienta sea funcional y pueda ser utilizada por el equipo de trabajo. Posteriormente se llevó a cabo una sensibilización al trabajo en equipo mediante el uso de Scrum y sobre el manejo de Kunagi para adquirir dominio y poder registrar la información del proyecto.

Una vez realizadas las anteriores actividades se procede a aplicar la propuesta que consiste en 4 adopciones incrementales, que se desarrollaron una por cada *Sprint*, en las que se incorporan los diferentes elementos metodológicos de Scrum y *Peopleware*, cuya síntesis de resultados se puede observar en la tabla 50.

Tabla 50. Síntesis adopciones de la propuesta

| **Indicadores** | **Adopciones** | | | |
|---|---|---|---|---|
| | **Primera** | **Segunda** | **Tercera** | **Cuarta** |
| **No. de Historias de Usuario** | 3 | 3 | 3 | 3 |
| **Horas trabajadas** | 55 | 76 | 64 | 64 |
| **Revisión** | Se presentó una dificultad en la historia de usuario “Hoja de vida del estudiante” relacionada con la complejidad de la consulta para sacar el promedio de notas de un estudiante por año. Este hecho ha retrasado el trabajo impidiendo que se pueda finalizar la historia de usuario, la cual debe ser | Se sigue presentando la misma dificultad en la historia de usuario “Hoja de vida del estudiante” detectada en el anterior *Sprint*,<br><br>La cual estaba relacionada con la complejidad de la consulta para sacar el promedio de notas de un estudiante por año. Este hecho ha retrasado el trabajo impidiendo que se pueda finalizar la historia de usuario y se pueda trabajar en las demás programadas para este *Sprint*. Por consiguiente | En este *Sprint* el trabajo se centró en la historia de usuario “Hoja de vida del estudiante”, la cual se trasladó del sprint número 1, al sprint número 2 y posteriormente al 3, debido a que presentaba bastante dificultad, por lo tanto se realizó una sesión de MobProgramming para determinar una posible solución a esta historia, se determinaron 2 alternativas que fueron: | En este *Sprint* se avanzó en relación con los anteriores, debido a que el trabajo ya realizado en las historias de usuario de cada uno de ellos, sirvió como base para el desarrollo de las tareas. Adicional a esto fue necesario a través de sesiones de Coding Dojo trabajar sobre la herramienta de creación de reportes JasperReports, ya que las historias de usuario se centraban en la presentación de reportes. |

Tabla 50. (Continuación)

| Indicadores | Adopciones | | | |
|---|---|---|---|---|
| | **Primera** | **Segunda** | **Tercera** | **Cuarta** |
| **Revisión** | trasladada al siguiente sprint. | será trasladada al siguiente sprint. | 1. Realizar la consulta a través de procedimientos almacenados en el gestor de base de datos PostgresSQL.<br>2. Subdividir en consulta más pequeñas y luego unirlas a través de programación en el Framework Angular JS.<br>Al ejecutar la primera alternativa se logró el objetivo pero los tiempos de ejecución de la consulta no eran óptimos, por lo tanto se descartó esta opción.<br>En cuanto a la segunda alternativa también se logró el objetivo, pero surgió un problema en la sincronización del framework y por | En este *Sprint* no se pudo terminar la historia de Usuario "Boletines de un Curso" debido a la dificultad de la consulta SQL y a interrupciones relacionadas con la dinámica de trabajo del Centro de Informática como capacitaciones y problemas técnicos de la red de datos. |

Tabla 50. (Continuación)

| Indicadores | Adopciones | | | |
|---|---|---|---|---|
| | **Primera** | **Segunda** | **Tercera** | **Cuarta** |
| | | | consiguiente los resultados eran variables.<br>Por las razones anteriores se determinó en el grupo de desarrollo, posponer el desarrollo de esta historia de usuario, para realizar un análisis más detallado posteriormente y así proseguir con las historias de usuario seleccionadas para este *Sprint* y no retrasar el desarrollo del proyecto. | |
| **Retrospectiva** | | Después de hacer un autoanálisis de la forma como se está trabajando el equipo determinó redefinir el *done* y suspender las tareas de elaborar y aprobar mockups, debido a que en el sprint | | En la reunión realizada por el equipo sobre cómo se estaba trabajando se determinó continuar con la tareas del Done sin hacer modificaciones, ya que las historias se enfocan en la presentación de reportes. Se |

Tabla 50. (Continuación)

| Indicadores | Adopciones | | | |
|---|---|---|---|---|
| | **Primera** | **Segunda** | **Tercera** | **Cuarta** |
| | | siguiente se trabajará únicamente en reportes. También se deja de utilizar el framework Sequelize, debido a que la complejidad de la consulta no lo permite; por lo tanto se va a utilizar simplemente SQL y se dará continuidad a las tareas:<br>- Codificar las Historias de Usuario.<br>- Elaborar casos de prueba.<br>- Ejecutar casos de prueba.<br><br>Por otra parte el equipo concluyó que habían cosas que se debían hacer y no se estaban realizando como:<br>- *Daily meeting*<br>- Planear y evaluar el trabajo. | | planteó seguir utilizando las tecnicas MobProgramming, Coding Dojo, Pomodoro, Niko Niko, Kanban, con el objetivo de conocerlas mejor.<br><br>Se propuso utilizar más el Burdown Chart como elemento que permite visualizar la complejidad manejada por el equipo, para soportar la toma de decisiones a tiempo y evitar que haya retrasos en el desarrollo del proyecto.<br>Calcular la complejidad que tiene cada historia de usuario. |

Tabla 50. (Continuación)

| Indicadores | Adopciones | | | |
|---|---|---|---|---|
| | **Primera** | **Segunda** | **Tercera** | **Cuarta** |
| | | - Identificar una técnica que contribuya en dar solución a la historia de usuario que ha generado dificultades debido a su complejidad. | | |
| **Etapa - Actividad** | Se realizó:<br>- *Sprint planning meeting*.<br>- *Daily meeting*.<br>- Definición del *Done.* | Se realizó:<br>- *Sprint planning meeting*.<br>- *Daily meeting*.<br>- Reestructuración del *Done.*<br>- *Sprint* review.<br>- Retrospectiva. | Se realizó:<br>- *Sprint planning meeting*.<br>- *Daily meeting*.<br>- *Sprint* review. | Se realizó:<br>- *Sprint planning meeting*.<br>- *Daily meeting*.<br>- *Sprint* review.<br>- Retrospectiva. |
| **Rol** | Se definió el rol:<br>- *Product Owner*. | Se definió el rol:<br>- Scrum Master. | Se trabajó con:<br>- *Team Developer*. | Se trabajó con:<br>- *Team Developer*. |
| **Artefacto** | Se elaboró:<br>- *Product Backlog*.<br>- *Sprint Backlog*. | Se actualizó:<br>- *Product Backlog*.<br>- *Sprint Backlog*. | Se actualizó:<br>- *Product Backlog*.<br>- *Sprint Backlog*.<br>- Métricas. | Se actualizó:<br>- *Product Backlog*.<br>- *Sprint Backlog*. |

Tabla 50. (Continuación)

| Indicadores | Adopciones | | | |
|---|---|---|---|---|
| | **Primera** | **Segunda** | **Tercera** | **Cuarta** |
| **Herramienta** | Se instaló y configuró la herramienta:<br><br>- Kunagi | Se utilizó:<br>- El tablero Kanban incluido dentro de Kunagi. | Se utilizó:<br><br>- Tomato-timer que es una herramienta web, que facilita la aplicación de la técnica Pomodoro. | Se utilizó:<br><br>- Kunagi para continuar con la gestión del *Product Backlog*, *Sprint Backlog*, tablero Kanban para administrar las tareas del *Done* y *Burndown Chart* como elemento gráfico para el análisis de la complejidad por historia de usuario. |
| **Técnica** | | Kanban posibilita visualizar rápidamente las tareas definidas en el *Done*, clasificándolas por pendientes, en ejecución y terminadas, facilitando el seguimiento a los integrantes del development team.<br>- Niko Niko para registrar el estado de ánimo de los | Se utilizaron las técnicas:<br><br>- Pomodoro con el objetivo de mejorar la administración del tiempo en el desarrollo de las tareas del *Done*.<br>- Se continúa trabajando con Niko Niko. | Se utilizaron dos técnicas:<br><br>- Mob programming que es una estrategia de desarrollo en el cual el *development team* trabaja sobre una tarea o actividad concreta y compleja del proyecto al mismo tiempo en un mismo ordenador. |

Tabla 50. (Continuación)

| Indicadores | Adopciones | | | |
|---|---|---|---|---|
| | **Primera** | **Segunda** | **Tercera** | **Cuarta** |
| | | integrantes del *development team.* | | - Coding Dojo en la cual cada integrante del *development team* hace sus aportes de acuerdo a sus conocimientos indiferentemente del nivel que tenga, permitiendo al grupo adquirir nuevas habilidades en técnicas y tecnologías de manera colaborativa y no competitiva. |
| **Lineamiento** | | | Se utilizó:<br><br>- *Burndown Chart* para observar de manera gráfica la complejidad manejada por el equipo en cada *Sprint*. | Se utilizó:<br><br>- Reporte de velocidad y tiempo restante de trabajo |

Fuente: La presente investigación 2017

A continuación se presenta un análisis del desarrollo por cada *Sprint* basado en el *Burdown Chart*.

En la figura 21 se observa que el equipo de desarrollo al inicio del sprint tuvo un buen ritmo de trabajo, cumpliendo con la tareas del *Done* previamente definidas para cada historia de usuario, a pesar de que hay espacios en donde la línea es constante debido a que esos días no fueron laborables, la gráfica en la mayoría del tiempo se mantuvo por debajo de la media, pero casi al final se sobrepasa debido a que la estimación de la historia de usuario “Hoja de vida del Estudiante” no fue correcta y tuvo una complejidad mayor, por consiguiente fue necesario trasladarla al siguiente *Sprint* para terminar las tareas pendientes.

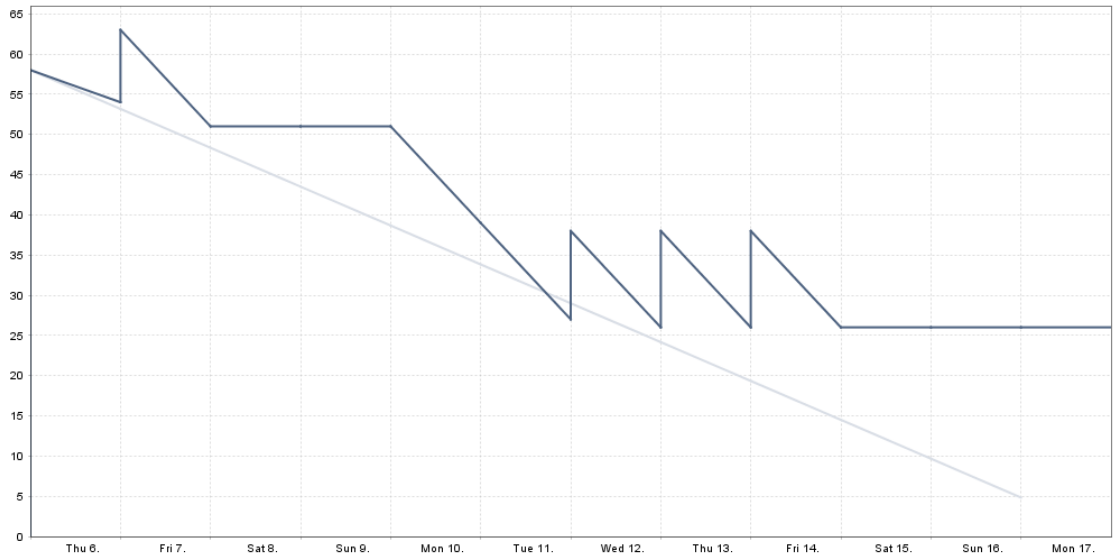
Figura 20. *Burndown Chart Sprint* No. 1


Fuente: La presente investigación 2017

Continuando con el segundo incremento, la figura 22 indica que la complejidad manejada es mayor lo cual hace de que la gráfica siempre este por encima de la media y los tiempos definidos se estén redefiniendo constantemente, demostrando errores graves de estimación por parte del *Development Team*, quizá debido a la dificultad para determinar cuándo se terminarán las tareas, de ahí la importancia de aplicar el *Daily meeting* sugerido en la retrospectiva de este *Sprint* como algo que

se debía hacer y no se estaba realizando. Además, de mejorar la planeación y evaluación del trabajo por parte del equipo.

Figura 21. *Burndown Chart Sprint* No. 2

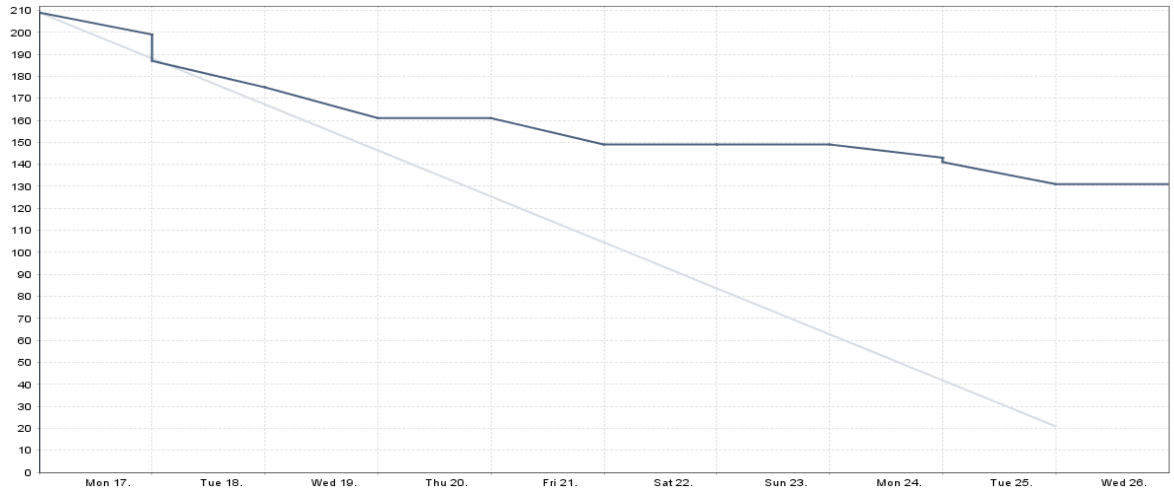



Fuente: La presente investigación 2017

Evaluando el tercer *Sprint*, la figura 23 muestra que las historias trabajadas siguen siendo complejas especialmente la historia “Hoja de vida del Estudiante”, la cual ya viene desde el *Sprint* 1 hasta el *Sprint* 3, en donde el *Development Team* para finalizar sus tareas, se apoyó en la técnica MobProgramming realizando sesiones de trabajo en las que se generaron alternativas que daban solución, pero no eran lo bastante eficientes como para implementarlas y dar por terminada la historia; por lo que se resolvió postergarla y continuar con las demás, decisión tomada casi al final del *Sprint*, de ahí que la gráfica indique un desfase importante en relación con el tiempo ideal propuesto por el grupo en la planificación del *Sprint*.

En el último *Sprint*, la figura 24 permite visualizar que al inicio el trabajo era equilibrado y estaba por debajo de la media, lo cual es el reflejo de la terminación de las tareas de la historia “Cuadro de honor de Estudiantes”, pero al iniciar con la historia “Boletines de un Curso” nuevamente hay dificultades con las consultas SQL las cuales son complejas y retrasan notablemente el trabajo del equipo. Adicionalmente, se suman interrupciones que no se habían contemplado como capacitaciones y problemas con la red de datos.

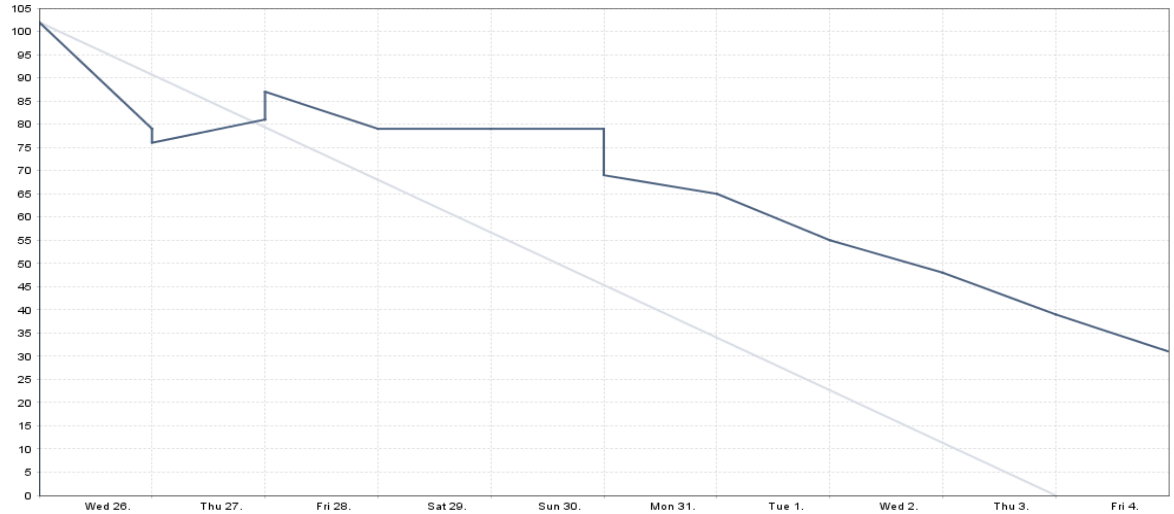
Figura 22. *Burndown Chart Sprint* No. 3




Fuente: La presente investigación 2017

Lo anterior hizo que solo se pudiera terminar una de las tres historias de usuario de la pila del *Sprint* y avanzar un 54% en la segunda, por consiguiente será necesario trasladarla junto con la tercera al siguiente *Sprint*, para el cual según la retrospectiva se mantienen las tareas del *Done* y la aplicación de las diferentes técnicas (MobProgramming, Coding Dojo, Niko Niko y Pomodoro) para empoderamiento y mejora de la productividad del *Development Team*.

Figura 23. *Burndown Chart Sprint* No. 4




Fuente: La presente investigación 2017

# 3. CONCLUSIONES

En el Centro de Informática de la Universidad de Nariño predomina el trabajo individual, en la realización del proceso de desarrollo, mantenimiento y soporte de productos software. El trabajo, se centra en la ejecución de actividades; y en un nivel muy bajo, se hacen acciones de planeación y evaluación. Así mismo, no se soporta la gestión del proceso mediante el uso de herramientas computacionales. En este sentido, no se logra evidenciar que exista una forma o método estandarizado y explicito que propicie el trabajo en equipo.

Al realizar el análisis comparativo de los elementos metodológicos del proceso de software del Centro de informática y los definidos por Scrum, se encuentra como similitudes, la especificación de requerimientos de software como parte de la pila de producto, la codificación como una actividad del Done en un *Sprint*; y el rol de desarrollador como integrante del *Development team*. La diferencia más relevante corresponde a la ausencia de métricas que posibiliten evaluar el proceso de desarrollo, mantenimiento y soporte de productos software.

Al evaluar las herramientas Kunagi, IceScrum, GanttProject, Scrumpy y Sprintometer utilizando el modelo OAM-F/OSS, se obtuvo como resultado final que Kunagi es la herramienta con el ponderado más alto, resultado de la suma de valores parciales obtenidos en cada uno de los criterios evaluados.

Se plantea una propuesta de trabajo en equipo, basada en Scrum y *Peopleware*, soportada por la herramienta de software libre Kunagi, para el Centro de Informática de la Universidad de Nariño. Esta propuesta se compone por cuatro etapas de adopción incrementales, en las cuales, se incorporan los elementos metodológicos de Scrum y se incluye las técnicas de: Niko Niko para registrar el estado de ánimo del equipo y mejorar su motivación, Pomodoro con el fin de administrar el tiempo de manera eficiente y mejorar la productividad, Kanban para visualizar el trabajo en equipo, MobProgramming en la solución de problemas de manera colectiva y Coding Dojo que contribuye en la socialización de conocimiento entre los integrantes del equipo.

Un aspecto a tener en cuenta en la aplicación de la propuesta fue el impacto que esta podría tener en los participantes, debido a la generación de cambios en la forma de trabajo actual. En este sentido, la estructura de la propuesta de trabajo en equipo para el Centro de Informática de la Universidad de Nariño, se plantea en adopciones incrementales con el objetivo de no saturar con muchos elementos al equipo y permitir que éste se adapte gradualmente a la nueva forma de trabajo.

La utilización de la herramienta Kunagi fue fundamental en la aplicación de la propuesta, para la gestión del proceso de desarrollo de un producto software, incorporando elementos de Scrum, y posibilitando tomar decisiones a partir de la información generada por la herramienta.

En la apropiación de la propuesta de trabajo en equipo, se desarrollan las adopciones a través de cuatro sprints, presentando motivación, por parte de los integrantes del equipo, en el conocimiento, principalmente de las técnicas de *Peopleware*. En el proceso, se logra identificar que debe existir un avance por parte del equipo en la estimación de la complejidad de las historias de usuario.

Al aplicar la propuesta en una Universidad pública hubo factores de tipo contractual relacionados con el presupuesto, que influyeron de manera negativa en los tiempos de ejecución de las actividades de acuerdo con el cronograma fijado, donde fue necesario replantearlo.

# 4. RECOMENDACIONES

Se elaboró una propuesta para trabajo en equipo que incluye Scrum, y técnicas para motivar, socializar conocimiento y mejorar la productividad; la cual fue apropiada a través de cuatro adopciones por un equipo de trabajo de la Universidad de Nariño. En este sentido, se recomienda que desde la dirección del Centro de Informática, se apoye con la continuidad del trabajo realizado por el equipo; y además este sirva para replicar el conocimiento a los demás integrantes de la dependencia.

La propuesta de trabajo en equipo planteada en esta investigación, incluye técnicas como Niko Niko que posibilitan identificar el estado de ánimo y la motivación de los integrantes del equipo, como insumo para plantear alternativas de intervención. En este orden de ideas, se recomienda hacer una evaluación de herramientas de software libre que soporten el uso de la técnica.

Este trabajo investigativo presenta una propuesta para trabajo en equipo basada en Scrum y técnicas de *Peopleware*, que fue aplicada en el desarrollo de un producto software para el Centro de Informática de la Universidad de Nariño. Por esta razón, se recomienda como trabajo futuro, desarrollar una nueva investigación, dónde se determine cuál sería el impacto de la apropiación e implementación de la propuesta en la gestión del trabajo en equipo.

## BIBLIOGRAFÍA


Bannerman, P. L., Hossain, E., & Jeffery, R. (2012). Scrum practice mitigation of global software development coordination challenges: a distinctive advantage?. *System Science (HICSS), 2012 45th Hawaii International Conference on*, (págs. 5309-5318). Recuperado el 14 de 02 de 2016, de https://www.academia.edu/1309295/Scrum_Practice_Mitigation_of_Global_Software_Development_Coordination_Challenges_A_Distinctive_Advantage

Beck, K., & Y Otros. (2001). *Manifiesto ágil.* Obtenido de http://www.agilemanifesto.org

Brekkan, E., & Mathisen , E. (2010). Introducing Scrum in Companies in Norway: A Case Study. *In de Proceedings of Informing Science & IT Education Conference (InSITE)*, *2010*, págs. 331-351. Recuperado el 14 de 02 de 2016, de https://www.researchgate.net/profile/Eystein_Mathisen/publication/235769725_Introducing_Scrum_in_Companies_in_Norway_A_Case_Study/links/0912f513633c46e51b000000.pdf

Canós, J. H., Letelier, P., & Penadés, M. (2003). Métodologías Ágiles en el Desarrollo de Software. Universidad Politécnica de Valencia. Valencia, España. Recuperado el 01 de 04 de 2016, de http://ima.udg.edu/Docencia/07-08/3105200728/TodoAgil.pdf

Colla, P. (2012). Marco para evaluar el valor en metodología Scrum. *13th Argentine Symposium on Software Engineering.* La Plata-Argentina. Recuperado el 27 de 01 de 2016, de http://41jaiio.sadio.org.ar/sites/default/files/086_ASSE_2012.pdf

de Souza Bermejo, P. H., Zambalde, A. L., Olímpio, A., Souza, S. A., Zuppo, L. A., & Rosa, P. L. (2014). Agile principles and achievement of success in software development: A quantitative study in Brazilian organizations. Recuperado el 20 de 04 de 2016, de http://www.sciencedirect.com/science/article/pii/S2212017314002485

ECURED. (01 de 04 de 2016). *Ciclo de Vida del Software,.* Obtenido de http://www.ecured.cu/Portal:Inform%C3%A1tica/Software#cite_note-Cvida-6

GALLARDO, Y., & MORENO, A. (1999). Serie Aprender a investigar. Módulo 3: Recolección de la información. Bogotá: Instituto Colombiano para el fomento de la educación superior, ICFES.

García-Algarra, J., & González-Ordás, J. (2010). KOWLAN: Una experiencia Scrum en un entorno de innovación. Madrid, España. Recuperado el 24 de 02 de 2016, de https://www.researchgate.net/profile/Javier_Garcia-Algarra/publication/233859834_KOWLAN_Una_experiencia_Scrum_en_un_entorno_de_innovacin/links/0fcfd513de80c22a63000000.pdf

Hernández, G., Martínez, Á., Argote, I., & Coral, D. (2015). Metodología adaptativa basada en Scrum: Caso empresas de la Industria de Software en San Juan de Pasto-Colombia. *Revista Tecnológica-ESPOL*, 28.

HERNANDEZ, R. (2010). *Metodología de la Investigación.* México: McGraw Hill.

Martinez, N., Ramon, H., & Bertone, R. (2012). Aplicabilidad de Competisoft a partir de un método ágil como Scrum. In XVIII Congreso Argentino de Ciencias de la Computación. Argentina. Recuperado el 14 de 02 de 2016, de http://sedici.unlp.edu.ar/bitstream/handle/10915/23722/Documento_completo.pdf?sequence=1

May, M., Morales, Y., Marrufo, J., & Martín, M. (2013). Implementación de un sistema para el control de activos ISOPTEC, bajo el estándar ITIL y metodología ágil Scrum. *2*, 176-190. México. Recuperado el 10 de 03 de 2016, de http://www.ecorfan.org/handbooks/pdf/Ciencias%20de%20la%20Ingenieria%20y%20Tecnologia%20Handbook%20T-II.pdf#page=188

NAUR, P., & RANDELL, B. (1968). SOFTWARE ENGINEERING: Report on a conference sponsored by the NATO science committee. Garmisch, Germany : Scientific Affairs Division NATO.

Ovesen, N. (2013). Facilitating Problem-Based Learning in Teams with Scrum. In the 15th International Conference onEngineering and *Product* Design Education. 856. Glasgow, UK. Recuperado el 27 de 01 de 2016, de http://vbn.aau.dk/ws/files/80762497/812.pdf

PRESSMAN, R. S. (2003). *Ingenieria del Software, un enfoque Práctico. 5ed.* México: Mc Graw Hill.

PROYECTOSAGILES.ORG. (s.f.). *cómo gestionar proyectos con Scrum: ¿Qué es Scrum?* Recuperado el 02 de 04 de 2016, de https://proyectosagiles.org/que-es-Scrum/

RAMÍREZ RAMÍREZ , I. (s.f.). *Los diferentes paradigmas de investigación y su incidencia sobre los diferentes modelos de investigación didàctica.* Recuperado el 10 de 05 de 2016, de http://josefa.aprenderapensar.net/

Schwaber, K., & Sutherland, J. (2013). La guía de Scrum: Las Reglas del Juego. 3-5. Recuperado el 03 de 04 de 2016, de http://Scrumguides.org/docs/Scrumguide/v1/Scrum-Guide-ES.pdf#zoom=100

Scrum MANAGER. (2013). *Modelo original de Scrum para desarrollo de software*. Recuperado el 28 de 04 de 2016, de http://www.Scrummanager.net/bok/index.php?title=Modelo_original_de_Scrum_para_desarrollo_de_software

TAMAYO TAMAYO, M. (2005). *Aprender a investigar - módulo 2: La investigación.* Bogotá: Arfo Editores Ltda.

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2009). The agile requirements refinery: Applying Scrum principles to software product management.

## ANEXOS

**Anexo A.** Formato de Encuesta realizada a los funcionarios del Centro de Informática de la Universidad de Nariño.

**Universidad Autónoma de Bucaramanga**

MAESTRÍA EN SOFTWARE LIBRE

METODOLOGÍA ADAPTATIVA BASADA EN Scrum, APOYADA POR HERRAMIENTAS DE SOFTWARE LIBRE PARA EL CENTRO DE INFORMÁTICA DE LA UNIVERSIDAD DE NARIÑO

CUESTIONARIO

El propósito de este cuestionario es caracterizar los elementos metodológicos del proceso de desarrollo, soporte y mantenimiento de software del Centro de Informática de la Universidad de Nariño, con el fin de formular una propuesta de trabajo basada en Scrum.

**Elementos metodológicos.**

La metodología es un camino definido que debe guiar el proceso de hacer software y debe contar con elementos como: a.) Las etapas o fases a realizar, b.) Las actividades para alcanzar los objetivos, c.) Los roles de las personas que interactúan, d.) Los artefactos o entregables, e.) Las herramientas que soporten el proceso; y f.) Los principios o criterios que posibiliten la toma de decisiones.

Teniendo en cuenta la definición anterior conteste de manera objetiva y sincera las preguntas que se presentan a continuación:

**Descripción Socio-Demográfica**

1. ¿Cuántos años tiene? _____
2. ¿Genero? _____
3. ¿Hace cuánto tiempo (años-meses) trabaja en la dependencia? _____
4. ¿Cuál es el cargo que desempeña actualmente?

_____

5. ¿Hace cuánto tiempo (años-meses) viene desempeñando este cargo? _____

6. ¿Cuál es máximo nivel de formación que tiene?
Bachiller____técnico____tecnólogo____profesional____especialista____
maestría____doctorado y postdoctorado____

**Etapas o fases**

7. ¿Cuáles de las siguientes fases o etapas aplica para el desarrollo, soporte y mantenimiento de software? (puede seleccionar más de una opción):

___ Planeación de un nuevo desarrollo

___ Planeación del soporte y mantenimiento de productos software existentes

___ Organización de los recursos y actividades a realizar

___ Ejecución o desarrollo de las actividades

___ Seguimiento y control de las actividades que se están ejecutando o desarrollando

___ Evaluación de las actividades ejecutadas o desarrolladas

___ Otra (s), ¿Cuál (es)? _____

8. ¿Se plantea objetivos que se desea alcanzar, antes de iniciar el proceso de desarrollo de un nuevo producto, soporte y mantenimiento de software?

___ Si

___ No

___ No sabe / No responde

9. Si la respuesta anterior fue No, manifieste la razón por la que no se realiza

_____

**Actividades**

10. ¿Qué actividades lleva a cabo para la realización de un nuevo producto sofware?

_____

11. ¿Qué actividades lleva a cabo para realizar el soporte y mantenimiento de software?

_____

12. ¿Para el desarrollo de las actividades en el Centro de Informática, se utiliza una forma o método de trabajo?

___ Si Cuál _____

___ No

___ No sabe / No responde

**Roles:** Tendencia de un integrante de un equipo o grupo a comportarse, colaborar, interrelacionarse con otros miembros de una forma particular.

13. ¿Desempeña algún rol en la realización de las actividades de desarrollo, soporte o mantenimiento de software?

___ Si ¿Cuál?_____
___ No

**Entregables:** Son los productos que se elaboran durante el proceso de construcción, soporte o mantenimiento de software (Diagramas, documentos, listas de inspección, etc) y hacen parte de la documentación

14. Describa los entregables que elabora en el ejercicio de su función.

| Actividad | Entregable(s) |
|---|---|
| Nuevo Desarrollo | |
| Soporte | |
| Mantenimiento | |

**Herramientas:** Software que sirve de apoyo al proceso de desarrollo, soporte y mantenimiento de software. (framework, generadores de documentación, control de versiones, control de errores, planificar proyectos etc.)

15. Describa las herramientas que utiliza en el ejercicio de su función.

| Actividad | Herramienta |
|---|---|
| Nuevo Desarrollo | |

| Actividad | Herramienta |
|---|---|
| Soporte | |
| Mantenimiento | |

**Lineamientos**

16. ¿Se hace uso de indicadores, factores o métricas para tomar decisiones en el proceso de desarrollo, soporte y mantenimiento de software?

___ Si
___ No
___ No sabe / No responde

Si la respuesta anterior fue No, manifieste la razón por la cual no se realiza y vaya al final de la encuesta.

_____
_____
_____

17. ¿Si la respuesta a la anterior pregunta fue Si, qué indicadores, factores o métricas utiliza en el ejercicio de su función? (puede seleccionar más de una opción):

___ Cumplimiento de objetivos

___ Desempeño en el trabajo

___ Velocidad de trabajo

___ LOC (líneas de código)/hora

___ Complejidad del trabajo realizado

___ Tiempo de respuesta a correcciones

___ Trabajo culminado

___ Trabajo por culminar

___ Otro(s), ¿Cuál (es)?: _____

_____

18. ¿Dónde se almacena la información de los indicadores, factores o métricas?

Gracias por su colaboración.

**Anexo B.** Instrumento de evaluación herramientas de Software Libre para la gestión del proceso Scrum.

**Universidad Autónoma de Bucaramanga**
MAESTRÍA EN SOFTWARE LIBRE
EVALUACIÓN DE HERRAMIENTAS DE SOFTWARE LIBRE PARA LA GESTIÓN DE PROYECTOS EN Scrum
CUESTIONARIO

## ACEPTACIÓN/USABILIDAD

### Usabilidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Existen en el programa indicadores y alertas de progreso y error? | | | | | |
| 2. | ¿Es posible trabajar en el aplicativo solo haciendo uso del teclado? | | | | | |
| 3. | ¿Cuál es el nivel de ayuda que posee el programa, dónde expliquen el funcionamiento de sus opciones? | | | | | |

### Interfaz de usuario

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tan intuitiva es la interfaz de usuario? | | | | | |
| 2. | ¿Qué tan fácil son de relacionar los menús e iconos con las acciones a realizar? | | | | | |
| 3. | ¿Qué tan fácil es de configurar el aplicativo en un idioma diferente? | | | | | |
| 4. | ¿Qué tan configurable es la apariencia del aplicativo? | | | | | |

## Implementación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tanto conocimiento técnico se necesita para su instalación? | | | | | |
| 2. | ¿Cuál es el porcentaje de tiempo para la instalación? | | | | | |
| 3. | ¿Cuál es el número necesario de otras herramientas para la instalación del software? | | | | | |

**ADMINISTRACIÓN**

## Actividad de versiones

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de control para las versiones del producto? | | | | | |

## Relación entre *stakeholders*

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de colaboración que permite la herramienta entre los integrantes del equipo? | | | | | |
| 2. | ¿Cuál es el nivel de interacción que permite el aplicativo de los integrantes del equipo? | | | | | |

**EFICIENCIA**

## Seguridad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de usuarios? | | | | | |
| 2. | ¿Qué nivel de facilidad ofrece para la gestión de grupos? | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 3. | ¿Qué nivel de facilidad ofrece para la gestión de permisos? | | | | | |

## Fiabilidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de recuperación de los datos tiene ante la presencia de fallas? | | | | | |
| 2. | ¿Cuál es el nivel de respuesta en la ejecución de las acciones críticas del software? | | | | | |
| 3. | ¿Qué tan estable es la herramienta al momento de trabajar? | | | | | |

# ENTRENAMIENTO (CAPACITACIÓN/DOCUMENTACIÓN)

## Documentación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de documentación está disponible en la web? | | | | | |
| 2. | ¿Cuál es el nivel de actualización de la documentación existente con relación a la última versión del software? | | | | | |

## Capacitación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el porcentaje de tiempo que se requiere capacitar a una persona para usar el software? | | | | | |

# INTEGRACIÓN

## Colaboración con otros productos

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de integración que permite con otras aplicaciones? | | | | | |

## PORTABILIDAD

### Independencia de plataforma

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿En cuántas plataformas se puede instalar la herramienta? | | | | | |
| 2. | ¿En cuántos ambientes (Escritorio, WEB, Móvil, Servicio WEB y consola) se puede usar la herramienta? | | | | | |

## SOFTWARE/PRODUCTO

### Soporte

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de fallas? | | | | | |
| 2. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de inquietudes? | | | | | |

### Edad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de madurez del producto software? | | | | | |
| 2. | ¿Cuál es el nivel de frecuencia con el que se actualiza el software? | | | | | |

### Funcionalidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de parametrización tiene el software? | | | | | |

**ESPECIFICIDAD**

## Administración

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de roles? | | | | | |

## Contenido

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la identificación de las necesidades del cliente para la construcción del product *backlog*? | | | | | |
| 2. | ¿Con qué facilidad se puede construir el sprint *backlog*? | | | | | |
| 3. | ¿Con qué facilidad se puede priorizar las necesidades del product *backlog*? | | | | | |
| 4. | ¿Cuál es el nivel de facilidad que ofrece para la planeación del sprint *planning meeting*? | | | | | |
| 5. | ¿Cuál es el nivel de facilidad que ofrece para el desarrollo del sprint *planning meeting*? | | | | | |
| 6. | ¿Cuál es el nivel de facilidad que ofrece para la evaluación del sprint *planning meeting*? | | | | | |
| 7. | ¿Qué nivel de facilidad ofrece para el desarrollo del sprint? | | | | | |
| 8. | ¿Cuál es el nivel de facilidad que ofrece para la realización del sprint review? | | | | | |
| 9. | ¿Con qué facilidad se puede realizar la retrospectiva? | | | | | |
| 10. | ¿Cuál es el nivel para el registro de métricas? | | | | | |
| 11. | ¿Qué nivel de facilidad ofrece para la recopilación de datos relacionados con las métricas? | | | | | |
| 12. | ¿Qué nivel la información ofrece a partir de los datos recopilados en las métricas? | | | | | |
| 13. | ¿Con qué facilidad se pueden definir el Done? | | | | | |
| 14. | ¿Con qué facilidad se puede gestionar la información de las métricas, a partir de los datos recopilados en los diferentes Sprints? | | | | | |
| 15. | ¿Cuál es el nivel de facilidad con el que se puede gestionar visualmente las necesidades definidas en el product *backlog*? | | | | | |
| 16. | ¿Qué nivel de facilidad ofrece para la gestión del daily meeting? | | | | | |
| 17. | ¿Cuál es el nivel de facilidad que ofrece para asignar tiempo a las actividades del *done*? | | | | | |

| Flujo de trabajo |
|---|

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿La herramienta permite la creación de tareas por roles? | | | | | |
| 2. | ¿Se puede realizar actividades de trabajo colaborativo entre los roles? | | | | | |

| Complementos |
|---|

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la motivación de los integrantes del Scrum team? | | | | | |
| 2. | ¿Qué nivel de facilidad ofrece para identificar el estado de ánimo del Scrum team? | | | | | |
| 3. | ¿Qué nivel de facilidad ofrece para compartir conocimiento entre los integrantes del Scrum team? | | | | | |
| 4. | ¿Qué nivel de facilidad ofrece para solucionar problemas complejos entre los integrantes del Scrum team ? | | | | | |

**Anexo C.** Evaluaciones de las herramientas

**Evaluación herramienta Kunagi**

Fecha de evaluación: 17-04-2017
Evaluadores: Carlos Jose Arevalo Delgado, Diego Esteban Paredes Burbano y Geovany Steven Viteri Salazar
Lugar: Universidad Mariana

**ACEPTACIÓN/USABILIDAD**

| Usabilidad |
|---|

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Existen en el programa indicadores y alertas de progreso y error? | | | X | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2. | ¿Es posible trabajar en el aplicativo solo haciendo uso del teclado? | | | | | X |
| 3. | ¿Cuál es el nivel de ayuda que posee el programa, dónde expliquen el funcionamiento de sus opciones? | X | | | | |

### Interfaz de usuario

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tan intuitiva es la interfaz de usuario? | | X | | | |
| 2. | ¿Qué tan fácil son de relacionar los menús e iconos con las acciones a realizar? | X | | | | |
| 3. | ¿Qué tan fácil es de configurar el aplicativo en un idioma diferente? | | | | | X |
| 4. | ¿Qué tan configurable es la apariencia del aplicativo? | | | | | X |

### Implementación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tanto conocimiento técnico se necesita para su instalación? | | | | | X |
| 2. | ¿Cuál es el porcentaje de tiempo para la instalación? | | | | | X |
| 3. | ¿Cuál es el número necesario de otras herramientas para la instalación del software? | | | | X | |

## ADMINISTRACIÓN

### Actividad de versiones

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de control para las versiones del producto? | X | | | | |

### Relación entre *stakeholders*

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de colaboración que permite la herramienta entre los integrantes del equipo? | X | | | | |
| 2. | ¿Cuál es el nivel de interacción que permite el aplicativo de los integrantes del equipo? | X | | | | |

## EFICIENCIA

### Seguridad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de usuarios? | X | | | | |
| 2. | ¿Qué nivel de facilidad ofrece para la gestión de grupos? | X | | | | |
| 3. | ¿Qué nivel de facilidad ofrece para la gestión de permisos? | X | | | | |

### Fiabilidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de recuperación de los datos tiene ante la presencia de fallas? | X | | | | |
| 2. | ¿Cuál es el nivel de respuesta en la ejecución de las acciones críticas del software? | | X | | | |
| 3. | ¿Qué tan estable es la herramienta al momento de trabajar? | | X | | | |

## ENTRENAMIENTO (CAPACITACIÓN/DOCUMENTACIÓN)

### Documentación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de documentación está disponible en la web? | X | | | | |
| 2. | ¿Cuál es el nivel de actualización de la documentación existente con relación a la última versión del software? | X | | | | |

### Capacitación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el porcentaje de tiempo que se requiere capacitar a una persona para usar el software? | | | X | | |

## INTEGRACIÓN

### Colaboración con otros productos

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de integración que permite con otras aplicaciones? | | | | | X |

## PORTABILIDAD

### Independencia de plataforma

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿En cuántas plataformas se puede instalar la herramienta? | X | | | | |
| 2. | ¿En cuántos ambientes (Escritorio, WEB, Móvil, Servicio WEB y consola) se puede usar la herramienta? | | | | X | |

## SOFTWARE/PRODUCTO

### Soporte

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de fallas? | | | | X | |
| 2. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de inquietudes? | | | | X | |

### Edad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de madurez del producto software? | X | | | | |
| 2. | ¿Cuál es el nivel de frecuencia con el que se actualiza el software? | | | X | | |

### Funcionalidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de parametrización tiene el software? | X | | | | |

## ESPECIFICIDAD

### Administración

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de roles? | X | | | | |

### Contenido

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la identificación de las necesidades del cliente para la construcción del product *backlog*? | | | | | X |
| 2. | ¿Con que facilidad se puede construir el sprint *backlog*? | | X | | | |
| 3. | ¿Con que facilidad se puede priorizar las necesidades del product *backlog*? | X | | | | |
| 4. | ¿Cuál es el nivel de facilidad que ofrece para la planeación del sprint *planning meeting*? | X | | | | |
| 5. | ¿Cuál es el nivel de facilidad que ofrece para el desarrollo del sprint *planning meeting*? | | | | | X |
| 6. | ¿Cuál es el nivel de facilidad que ofrece para la evaluación del sprint *planning meeting*? | | X | | | |
| 7. | ¿Qué nivel de facilidad ofrece para el desarrollo del sprint? | X | | | | |
| 8. | ¿Cuál es el nivel de facilidad que ofrece para la realización del sprint review? | | X | | | |
| 9. | ¿Con que facilidad se puede realizar la retrospectiva? | | X | | | |
| 10. | ¿Cuál es el nivel para el registro de métricas? | | | | | X |
| 11. | ¿Qué nivel de facilidad ofrece para la recopilación de datos relacionados con las métricas? | | X | | | |
| 12. | ¿Qué nivel la información ofrece a partir de los datos recopilados en las métricas? | | | X | | |
| 13. | ¿Con que facilidad se pueden definir el Done? | X | | | | |
| 14. | ¿Con que facilidad se puede gestionar la información de las métricas, a partir de los datos recopilados en los diferentes Sprints? | | | | | X |
| 15. | ¿Cuál es el nivel de facilidad con el que se puede gestionar visualmente las necesidades definidas en el product *backlog*? | X | | | | |
| 16. | ¿Qué nivel de facilidad ofrece para la gestión del daily meeting? | | X | | | |
| 17. | ¿Cuál es el nivel de facilidad que ofrece para asignar tiempo a las actividades del *done*? | X | | | | |

## Flujo de trabajo

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿La herramienta permite la creación de tareas por roles? | X | | | | |
| 2. | ¿Se puede realizar actividades de trabajo colaborativo entre los roles? | X | | | | |

## Complementos

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la motivación de los integrantes del Scrum team? | | X | | | |
| 2. | ¿Qué nivel de facilidad ofrece para identificar el estado de ánimo del Scrum team? | | | X | | |
| 3. | ¿Qué nivel de facilidad ofrece para compartir conocimiento entre los integrantes del Scrum team? | | X | | | |
| 4. | ¿Qué nivel de facilidad ofrece para solucionar problemas complejos entre los integrantes del Scrum team? | | X | | | |

**Evaluación herramienta IceScrum**

Fecha de evaluación: 19-04-2017
Evaluadores: Carlos Jose Arevalo Delgado, Diego Esteban Paredes Burbano y Geovany Steven Viteri Salazar
Lugar: Universidad Mariana

## ACEPTACIÓN/USABILIDAD

### Usabilidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Existen en el programa indicadores y alertas de progreso y error? | | | X | | |
| 2. | ¿Es posible trabajar en el aplicativo solo haciendo uso del teclado? | | | | | X |
| 3. | ¿Cuál es el nivel de ayuda que posee el programa, dónde expliquen el funcionamiento de sus opciones? | | | | X | |

### Interfaz de usuario

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tan intuitiva es la interfaz de usuario? | X | | | | |
| 2. | ¿Qué tan fácil son de relacionar los menús e iconos con las acciones a realizar? | X | | | | |
| 3. | ¿Qué tan fácil es de configurar el aplicativo en un idioma diferente? | X | | | | |
| 4. | ¿Qué tan configurable es la apariencia del aplicativo? | | | X | | |

### Implementación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tanto conocimiento técnico se necesita para su instalación? | X | | | | |
| 2. | ¿Cuál es el porcentaje de tiempo para la instalación? | X | | | | |
| 3. | ¿Cuál es el número necesario de otras herramientas para la instalación del software? | | | | X | |

## ADMINISTRACIÓN

### Actividad de versiones

| No | Pregunta | MA | A | NA | B | MB |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | NB | | |
| 1. | ¿Cuál es el nivel de control para las versiones del producto? | X | | | | |

Relación entre *stakeholders*

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de colaboración que permite la herramienta entre los integrantes del equipo? | X | | | | |
| 2. | ¿Cuál es el nivel de interacción que permite el aplicativo de los integrantes del equipo? | X | | | | |

## EFICIENCIA

Seguridad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de usuarios? | X | | | | |
| 2. | ¿Qué nivel de facilidad ofrece para la gestión de grupos? | X | | | | |
| 3. | ¿Qué nivel de facilidad ofrece para la gestión de permisos? | X | | | | |

Fiabilidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de recuperación de los datos tiene ante la presencia de fallas? | X | | | | |
| 2. | ¿Cuál es el nivel de respuesta en la ejecución de las acciones críticas del software? | X | | | | |
| 3. | ¿Qué tan estable es la herramienta al momento de trabajar? | X | | | | |

## ENTRENAMIENTO (CAPACITACIÓN/DOCUMENTACIÓN)

Documentación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de documentación está disponible en la web? | X | | | | |
| 2. | ¿Cuál es el nivel de actualización de la documentación existente con relación a la última versión del software? | | | X | | |

Capacitación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el porcentaje de tiempo que se requiere capacitar a una persona para usar el software? | | | | X | |

## INTEGRACIÓN

### Colaboración con otros productos

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de integración que permite con otras aplicaciones? | | | | | X |

## PORTABILIDAD

### Independencia de plataforma

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿En cuántas plataformas se puede instalar la herramienta? | X | | | | |
| 2. | ¿En cuántos ambientes (Escritorio, WEB, Móvil, Servicio WEB y consola) se puede usar la herramienta? | | | | | X |

## SOFTWARE/PRODUCTO

### Soporte

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de fallas? | | | X | | |
| 2. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de inquietudes? | | | X | | |

### Edad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de madurez del producto software? | X | | | | |
| 2. | ¿Cuál es el nivel de frecuencia con el que se actualiza el software? | X | | | | |

### Funcionalidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de parametrización tiene el software? | X | | | | |

## ESPECIFICIDAD

### Administración

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de roles? | X | | | | |

### Contenido

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la identificación de las necesidades del cliente para la construcción del product *backlog*? | | X | | | |
| 2. | ¿Con que facilidad se puede construir el sprint *backlog*? | X | | | | |
| 3. | ¿Con que facilidad se puede priorizar las necesidades del product *backlog*? | | X | | | |
| 4. | ¿Cuál es el nivel de facilidad que ofrece para la planeación del sprint *planning meeting*? | | | | | X |
| 5. | ¿Cuál es el nivel de facilidad que ofrece para el desarrollo del sprint *planning meeting*? | | | | | X |
| 6. | ¿Cuál es el nivel de facilidad que ofrece para la evaluación del sprint *planning meeting*? | | | | | X |
| 7. | ¿Qué nivel de facilidad ofrece para el desarrollo del sprint? | X | | | | |
| 8. | ¿Cuál es el nivel de facilidad que ofrece para la realización del sprint review? | | | | | X |
| 9. | ¿Con que facilidad se puede realizar la retrospectiva? | | | X | | |
| 10. | ¿Cuál es el nivel para el registro de métricas? | | | | | X |
| 11. | ¿Qué nivel de facilidad ofrece para la recopilación de datos relacionados con las métricas? | | X | | | |
| 12. | ¿Qué nivel la información ofrece a partir de los datos recopilados en las métricas? | X | | | | |
| 13. | ¿Con que facilidad se pueden definir el Done? | | X | | | |
| 14. | ¿Con que facilidad se puede gestionar la información de las métricas, a partir de los datos recopilados en los diferentes Sprints? | | X | | | |
| 15. | ¿Cuál es el nivel de facilidad con el que se puede gestionar visualmente las necesidades definidas en el product *backlog*? | X | | | | |
| 16. | ¿Qué nivel de facilidad ofrece para la gestión del daily meeting? | | | | | X |
| 17. | ¿Cuál es el nivel de facilidad que ofrece para asignar tiempo a las actividades del *done*? | X | | | | |

### Flujo de trabajo

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿La herramienta permite la creación de tareas por roles? | X | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2. | ¿Se puede realizar actividades de trabajo colaborativo entre los roles? | | X | | | |

## Complementos

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la motivación de los integrantes del Scrum team? | | | | | X |
| 2. | ¿Qué nivel de facilidad ofrece para identificar el estado de ánimo del Scrum team? | | | | | X |
| 3. | ¿Qué nivel de facilidad ofrece para compartir conocimiento entre los integrantes del Scrum team? | | | | | X |
| 4. | ¿Qué nivel de facilidad ofrece para solucionar problemas complejos entre los integrantes del Scrum team? | | | | | X |

### Evaluación herramienta Sprintometer

Fecha de evaluación: 19-04-2017
Evaluadores: Carlos Jose Arevalo Delgado, Diego Esteban Paredes Burbano y Geovany Steven Viteri Salazar
Lugar: Universidad Mariana

### ACEPTACIÓN/USABILIDAD

## Usabilidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Existen en el programa indicadores y alertas de progreso y error? | | X | | | |
| 2. | ¿Es posible trabajar en el aplicativo solo haciendo uso del teclado? | | | X | | |
| 3. | ¿Cuál es el nivel de ayuda que posee el programa, dónde expliquen el funcionamiento de sus opciones? | | | X | | |

## Interfaz de usuario

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tan intuitiva es la interfaz de usuario? | | | X | | |
| 2. | ¿Qué tan fácil son de relacionar los menús e iconos con las acciones a realizar? | | | X | | |
| 3. | ¿Qué tan fácil es de configurar el aplicativo en un idioma diferente? | | | | | X |
| 4. | ¿Qué tan configurable es la apariencia del aplicativo? | | | | | X |

### Implementación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tanto conocimiento técnico se necesita para su instalación? | | | | | X |
| 2. | ¿Cuál es el porcentaje de tiempo para la instalación? | | | | | X |
| 3. | ¿Cuál es el número necesario de otras herramientas para la instalación del software? | | | | | X |

## ADMINISTRACIÓN

### Actividad de versiones

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de control para las versiones del producto? | | | | | X |

### Relación entre *stakeholders*

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de colaboración que permite la herramienta entre los integrantes del equipo? | | | | | X |
| 2. | ¿Cuál es el nivel de interacción que permite el aplicativo de los integrantes del equipo? | | | | | X |

## EFICIENCIA

### Seguridad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de usuarios? | | | | | X |
| 2. | ¿Qué nivel de facilidad ofrece para la gestión de grupos? | | | | | X |
| 3. | ¿Qué nivel de facilidad ofrece para la gestión de permisos? | | | | | X |

### Fiabilidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de recuperación de los datos tiene ante la presencia de fallas? | | X | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2. | ¿Cuál es el nivel de respuesta en la ejecución de las acciones críticas del software? | X | | | | |
| 3. | ¿Qué tan estable es la herramienta al momento de trabajar? | | | X | | |

## ENTRENAMIENTO (CAPACITACIÓN/DOCUMENTACIÓN)

### Documentación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de documentación está disponible en la web? | X | | | | |
| 2. | ¿Cuál es el nivel de actualización de la documentación existente con relación a la última versión del software? | | | | X | |

### Capacitación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el porcentaje de tiempo que se requiere capacitar a una persona para usar el software? | X | | | | |

## INTEGRACIÓN

### Colaboración con otros productos

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de integración que permite con otras aplicaciones? | | | | | X |

## PORTABILIDAD

### Independencia de plataforma

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿En cuántas plataformas se puede instalar la herramienta? | | | | | X |
| 2. | ¿En cuántos ambientes (Escritorio, WEB, Móvil, Servicio WEB y consola) se puede usar la herramienta? | | | | | X |

## SOFTWARE/PRODUCTO

### Soporte

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| 1. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de fallas? | | | | X | |
| 2. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de inquietudes? | | | | X | |

### Edad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de madurez del producto software? | X | | | | |
| 2. | ¿Cuál es el nivel de frecuencia con el que se actualiza el software? | | | X | | |

### Funcionalidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de parametrización tiene el software? | | X | | | |

## ESPECIFICIDAD

### Administración

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de roles? | | | | | X |

### Contenido

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la identificación de las necesidades del cliente para la construcción del product *backlog*? | | | | | X |
| 2. | ¿Con que facilidad se puede construir el sprint *backlog*? | | X | | | |
| 3. | ¿Con que facilidad se puede priorizar las necesidades del product *backlog*? | | | | X | |
| 4. | ¿Cuál es el nivel de facilidad que ofrece para la planeación del sprint *planning meeting*? | | | | | X |
| 5. | ¿Cuál es el nivel de facilidad que ofrece para el desarrollo del sprint *planning meeting*? | | | | | X |
| 6. | ¿Cuál es el nivel de facilidad que ofrece para la evaluación del sprint *planning meeting*? | | | | | X |
| 7. | ¿Qué nivel de facilidad ofrece para el desarrollo del sprint? | | X | | | |
| 8. | ¿Cuál es el nivel de facilidad que ofrece para la realización del sprint review? | | | | | X |
| 9. | ¿Con que facilidad se puede realizar la retrospectiva? | | | | | X |
| 10. | ¿Cuál es el nivel para el registro de métricas? | | | | | X |

| | | | | | | |
|---|---|---|---|---|---|---|
| 11. | ¿Qué nivel de facilidad ofrece para la recopilación de datos relacionados con las métricas? | | X | | | |
| 12. | ¿Qué nivel la información ofrece a partir de los datos recopilados en las métricas? | X | | | | |
| 13. | ¿Con que facilidad se pueden definir el Done? | | X | | | |
| 14. | ¿Con que facilidad se puede gestionar la información de las métricas, a partir de los datos recopilados en los diferentes Sprints? | | | | X | |
| 15. | ¿Cuál es el nivel de facilidad con el que se puede gestionar visualmente las necesidades definidas en el product *backlog*? | | X | | | |
| 16. | ¿Qué nivel de facilidad ofrece para la gestión del daily meeting? | | | | | X |
| 17. | ¿Cuál es el nivel de facilidad que ofrece para asignar tiempo a las actividades del *done*? | | X | | | |

## Flujo de trabajo

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿La herramienta permite la creación de tareas por roles? | | | | | X |
| 2. | ¿Se puede realizar actividades de trabajo colaborativo entre los roles? | | | | | X |

## Complementos

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la motivación de los integrantes del Scrum team? | | | | | X |
| 2. | ¿Qué nivel de facilidad ofrece para identificar el estado de ánimo del Scrum team? | | | | | X |
| 3. | ¿Qué nivel de facilidad ofrece para compartir conocimiento entre los integrantes del Scrum team? | | | | | X |
| 4. | ¿Qué nivel de facilidad ofrece para solucionar problemas complejos entre los integrantes del Scrum team? | | | | | X |

**Evaluación herramienta Scrumpy**

Fecha de evaluación: 21-04-2017
Evaluadores: Carlos Jose Arevalo Delgado, Diego Esteban Paredes Burbano y Geovany Steven Viteri Salazar
Lugar: Universidad Mariana

**ACEPTACIÓN/USABILIDAD**

## Usabilidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Existen en el programa indicadores y alertas de progreso y error? | | X | | | |
| 2. | ¿Es posible trabajar en el aplicativo solo haciendo uso del teclado? | | | | X | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 3. | ¿Cuál es el nivel de ayuda que posee el programa, dónde expliquen el funcionamiento de sus opciones? | | | | | X |

### Interfaz de usuario

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tan intuitiva es la interfaz de usuario? | | | | | X |
| 2. | ¿Qué tan fácil son de relacionar los menús e iconos con las acciones a realizar? | | | X | | |
| 3. | ¿Qué tan fácil es de configurar el aplicativo en un idioma diferente? | | | | | X |
| 4. | ¿Qué tan configurable es la apariencia del aplicativo? | | | | | X |

### Implementación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tanto conocimiento técnico se necesita para su instalación? | | X | | | |
| 2. | ¿Cuál es el porcentaje de tiempo para la instalación? | | | X | | |
| 3. | ¿Cuál es el número necesario de otras herramientas para la instalación del software? | | | | | X |

## ADMINISTRACIÓN

### Actividad de versions

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de control para las versiones del producto? | X | | | | |

### Relación entre *stakeholders*

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de colaboración que permite la herramienta entre los integrantes del equipo? | | | | | X |
| 2. | ¿Cuál es el nivel de interacción que permite el aplicativo de los integrantes del equipo? | | | | | X |

## EFICIENCIA

### Seguridad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de usuarios? | | | | | X |
| 2. | ¿Qué nivel de facilidad ofrece para la gestión de grupos? | | | | | X |
| 3. | ¿Qué nivel de facilidad ofrece para la gestión de permisos? | | | | | X |

Fiabilidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de recuperación de los datos tiene ante la presencia de fallas? | | | | | X |
| 2. | ¿Cuál es el nivel de respuesta en la ejecución de las acciones críticas del software? | X | | | | |
| 3. | ¿Qué tan estable es la herramienta al momento de trabajar? | X | | | | |

## ENTRENAMIENTO (CAPACITACIÓN/DOCUMENTACIÓN)

Documentación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de documentación está disponible en la web? | X | | | | |
| 2. | ¿Cuál es el nivel de actualización de la documentación existente con relación a la última versión del software? | | | | X | |

Capacitación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el porcentaje de tiempo que se requiere capacitar a una persona para usar el software? | | | X | | |

## INTEGRACIÓN

Colaboración con otros productos

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de integración que permite con otras aplicaciones? | | | | | X |

## PORTABILIDAD

Independencia de plataforma

| No | Pregunta | MA | A | NA | B | MB |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | NB | | |
| 1. | ¿En cuántas plataformas se puede instalar la herramienta? | X | | | | |
| 2. | ¿En cuántos ambientes (Escritorio, WEB, Móvil, Servicio WEB y consola) se puede usar la herramienta? | | | | | X |

**SOFTWARE/PRODUCTO**

Soporte

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de fallas? | | | | X | |
| 2. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de inquietudes? | | | | X | |

Edad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de madurez del producto software? | X | | | | |
| 2. | ¿Cuál es el nivel de frecuencia con el que se actualiza el software? | | | | | X |

Funcionalidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de parametrización tiene el software? | | | | X | |

**ESPECIFICIDAD**

Administración

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de roles? | | | | | X |

Contenido

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la identificación de las necesidades del cliente para la construcción del product *backlog*? | | | | | X |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2. | ¿Con que facilidad se puede construir el sprint *backlog*? | X | | | | |
| 3. | ¿Con que facilidad se puede priorizar las necesidades del product *backlog*? | | | | X | |
| 4. | ¿Cuál es el nivel de facilidad que ofrece para la planeación del sprint *planning meeting*? | | | | | X |
| 5. | ¿Cuál es el nivel de facilidad que ofrece para el desarrollo del sprint *planning meeting*? | | | | | X |
| 6. | ¿Cuál es el nivel de facilidad que ofrece para la evaluación del sprint *planning meeting*? | | | | | X |
| 7. | ¿Qué nivel de facilidad ofrece para el desarrollo del sprint? | | | | X | |
| 8. | ¿Cuál es el nivel de facilidad que ofrece para la realización del sprint review? | | | | | X |
| 9. | ¿Con que facilidad se puede realizar la retrospectiva? | | | | | X |
| 10. | ¿Cuál es el nivel para el registro de métricas? | | | | | X |
| 11. | ¿Qué nivel de facilidad ofrece para la recopilación de datos relacionados con las métricas? | | | | X | |
| 12. | ¿Qué nivel la información ofrece a partir de los datos recopilados en las métricas? | | | | X | |
| 13. | ¿Con que facilidad se pueden definir el Done? | | | | | X |
| 14. | ¿Con que facilidad se puede gestionar la información de las métricas, a partir de los datos recopilados en los diferentes Sprints? | | | | | X |
| 15. | ¿Cuál es el nivel de facilidad con el que se puede gestionar visualmente las necesidades definidas en el product *backlog*? | | X | | | |
| 16. | ¿Qué nivel de facilidad ofrece para la gestión del daily meeting? | | | | | X |
| 17. | ¿Cuál es el nivel de facilidad que ofrece para asignar tiempo a las actividades del *done*? | | | | | X |

## Flujo de trabajo

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿La herramienta permite la creación de tareas por roles? | | | | | X |
| 2. | ¿Se puede realizar actividades de trabajo colaborativo entre los roles? | | | | | X |

## Complementos

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la motivación de los integrantes del Scrum team? | | | | | X |
| 2. | ¿Qué nivel de facilidad ofrece para identificar el estado de ánimo del Scrum team? | | | | | X |
| 3. | ¿Qué nivel de facilidad ofrece para compartir conocimiento entre los integrantes del Scrum team? | | | | | X |
| 4. | ¿Qué nivel de facilidad ofrece para solucionar problemas complejos entre los integrantes del Scrum team? | | | | | X |

**Evaluación herramienta GanttProject**

Fecha de evaluación: 24-04-2017
Evaluadores: Carlos Jose Arevalo Delgado, Diego Esteban Paredes Burbano y Geovany Steven Viteri Salazar
Lugar: Universidad Mariana

## ACEPTACIÓN/USABILIDAD

### Usabilidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Existen en el programa indicadores y alertas de progreso y error? | | | | X | |
| 2. | ¿Es posible trabajar en el aplicativo solo haciendo uso del teclado? | | | | | X |
| 3. | ¿Cuál es el nivel de ayuda que posee el programa, dónde expliquen el funcionamiento de sus opciones? | | | | | X |

### Interfaz de usuario

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tan intuitiva es la interfaz de usuario? | | X | | | |
| 2. | ¿Qué tan fácil son de relacionar los menús e iconos con las acciones a realizar? | X | | | | |
| 3. | ¿Qué tan fácil es de configurar el aplicativo en un idioma diferente? | X | | | | |
| 4. | ¿Qué tan configurable es la apariencia del aplicativo? | | | | X | |

### Implementación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué tanto conocimiento técnico se necesita para su instalación? | | | | | X |
| 2. | ¿Cuál es el porcentaje de tiempo para la instalación? | | | | | X |
| 3. | ¿Cuál es el número necesario de otras herramientas para la instalación del software? | | | | | X |

## ADMINISTRACIÓN

### Actividad de versiones

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de control para las versiones del producto? | | | | | X |

Relación entre *stakeholders*

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de colaboración que permite la herramienta entre los integrantes del equipo? | | | | | X |
| 2. | ¿Cuál es el nivel de interacción que permite el aplicativo de los integrantes del equipo? | | | | | X |

**EFICIENCIA**

Seguridad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de usuarios? | | | | X | |
| 2. | ¿Qué nivel de facilidad ofrece para la gestión de grupos? | | | | | X |
| 3. | ¿Qué nivel de facilidad ofrece para la gestión de permisos? | | | | | X |

Fiabilidad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de recuperación de los datos tiene ante la presencia de fallas? | | X | | | |
| 2. | ¿Cuál es el nivel de respuesta en la ejecución de las acciones críticas del software? | X | | | | |
| 3. | ¿Qué tan estable es la herramienta al momento de trabajar? | X | | | | |

**ENTRENAMIENTO (CAPACITACIÓN/DOCUMENTACIÓN)**

Documentación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de documentación está disponible en la web? | | | | X | |
| 2. | ¿Cuál es el nivel de actualización de la documentación existente con relación a la última versión del software? | | | | X | |

Capacitación

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el porcentaje de tiempo que se requiere capacitar a una persona para usar el software? | | | | X | |

## INTEGRACIÓN

### Colaboración con otros productos

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de integración que permite con otras aplicaciones? | | | | | X |

## PORTABILIDAD

### Independencia de plataforma

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿En cuántas plataformas se puede instalar la herramienta? | X | | | | |
| 2. | ¿En cuántos ambientes (Escritorio, WEB, Móvil, Servicio WEB y consola) se puede usar la herramienta? | | | | | X |

## SOFTWARE/PRODUCTO

### Soporte

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de fallas? | | | X | | |
| 2. | ¿Qué medios (Foro, Blog, Wiki, Preguntas frecuentes, Buzón de sugerencias) existen para la resolución de inquietudes? | | | X | | |

### Edad

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Cuál es el nivel de madurez del producto software? | | | | | X |
| 2. | ¿Cuál es el nivel de frecuencia con el que se actualiza el software? | | | | | X |

### Funcionalidad

| No | Pregunta | MA | A | NA | B | MB |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | NB | | |
| 1. | ¿Qué nivel de parametrización tiene el software? | | | | X | |

**ESPECIFICIDAD**

## Administración

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la gestión de roles? | | | X | | |

## Contenido

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la identificación de las necesidades del cliente para la construcción del product *backlog*? | | | | | X |
| 2. | ¿Con que facilidad se puede construir el sprint *backlog*? | | | | | X |
| 3. | ¿Con que facilidad se puede priorizar las necesidades del product *backlog*? | | | X | | |
| 4. | ¿Cuál es el nivel de facilidad que ofrece para la planeación del sprint *planning meeting*? | | | | | X |
| 5. | ¿Cuál es el nivel de facilidad que ofrece para el desarrollo del sprint *planning meeting*? | | | | | X |
| 6. | ¿Cuál es el nivel de facilidad que ofrece para la evaluación del sprint *planning meeting*? | | | | | X |
| 7. | ¿Qué nivel de facilidad ofrece para el desarrollo del sprint? | | | | | X |
| 8. | ¿Cuál es el nivel de facilidad que ofrece para la realización del sprint review? | | | | | X |
| 9. | ¿Con que facilidad se puede realizar la retrospectiva? | | | | | X |
| 10. | ¿Cuál es el nivel para el registro de métricas? | | | | | X |
| 11. | ¿Qué nivel de facilidad ofrece para la recopilación de datos relacionados con las métricas? | | | | X | |
| 12. | ¿Qué nivel la información ofrece a partir de los datos recopilados en las métricas? | | | | X | |
| 13. | ¿Con que facilidad se pueden definir el Done? | | X | | | |
| 14. | ¿Con que facilidad se puede gestionar la información de las métricas, a partir de los datos recopilados en los diferentes Sprints? | | | | | X |
| 15. | ¿Cuál es el nivel de facilidad con el que se puede gestionar visualmente las necesidades definidas en el product *backlog*? | X | | | | |
| 16. | ¿Qué nivel de facilidad ofrece para la gestión del daily meeting? | | | | | X |
| 17. | ¿Cuál es el nivel de facilidad que ofrece para asignar tiempo a las actividades del *done*? | | X | | | |

## Flujo de trabajo

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿La herramienta permite la creación de tareas por roles? | | | X | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2. | ¿Se puede realizar actividades de trabajo colaborativo entre los roles? | | | | | X |

## Complementos

| No | Pregunta | MA | A | NA NB | B | MB |
|---|---|---|---|---|---|---|
| 1. | ¿Qué nivel de facilidad ofrece para la motivación de los integrantes del Scrum team? | | | | | X |
| 2. | ¿Qué nivel de facilidad ofrece para identificar el estado de ánimo del Scrum team? | | | | | X |
| 3. | ¿Qué nivel de facilidad ofrece para compartir conocimiento entre los integrantes del Scrum team? | | | | | X |
| 4. | ¿Qué nivel de facilidad ofrece para solucionar problemas complejos entre los integrantes del Scrum team? | | | | | X |

**Anexo D.** Ventajas y Desventajas de las Herramientas Evaluadas.

Ventajas de Herramientas Evaluadas.

| | **Herramientas Evaluadas** | | | |
|---|---|---|---|---|
| **Criterio** | **IceScrum** | **GanttProject** | **Scrumpy** | **Sprintometer** |
| **Aceptación/ Usabilidad** | -Posee alertas de alerta y error<br>-Permite relacionar cada menú e icono con su correspondiente actividad<br>-Permite cambiar la herramienta a 10 diferentes idiomas<br>-Permite movilizar ítems dependiendo la necesidad | -Permite relacionar cada menú e icono con su correspondiente actividad<br>- Permite cambiar la herramienta a 10 diferentes idiomas<br>-Permite cambiar tamaño de letra y color de las graficas<br>- La instalación se realiza de manera ágil además no se requieren conocimientos técnicos | -Posee alertas de alerta y error<br>-Permite relacionar cada menú e icono con su correspondiente actividad | -Posee alertas de alerta y error<br>- La instalación se realiza de manera ágil además no se requieren conocimientos técnicos |
| **Administración** | -Posee un blog especializado por cada una de las versiones del product<br>-Permite la colaboración por parte de todo el equipo de trabajo simultáneamente<br>-Permite la interacción del equipo a través de chats<br>-Tiene opción de login | -Permite asignar tareas a los usuarios | -Contiene una página especializada para manejar las versiones correspondientes | - |
| **Eficiencia** | -Permite la gestión de roles, grupos y permisos de acuerdo a la metodología Scrum<br>-Guarda la información de los proyectos en una | -Guarda el proyecto en un archivo<br>-Presenta una opción de recuperación del proyecto anteriormente trabajado. | -Da respuesta inmediata frente a las acciones criticas<br>-Es estable a la hora de trabajar | -Da respuesta inmediata frente a las acciones criticas |

| | Herramientas Evaluadas | | | |
|---|---|---|---|---|
| **Criterio** | **IceScrum** | **GanttProject** | **Scrumpy** | **Sprintometer** |
| | base de datos creada en la instalación<br>-Da respuesta inmediata frente a las acciones criticas<br>-Es estable a la hora de trabajar | -Da respuesta inmediata frente a las acciones criticas<br>-Es estable a la hora de trabajar | | |
| **Entrenamiento** | -Existe un alto nivel de documentación especializada en la web como guías, blogs y paginas alternas.<br>-Tiene documentación actualizada frente a la última versión lanzada de la herramienta en la página oficial<br>-No demanda tiempo la capacitación a una persona ya que las actividades se realizan de manera intuitiva | -No demanda tiempo la capacitación a una persona ya que las actividades se realizan de manera intuitiva | -Existe un alto nivel de documentación especializada en la web como guías, blogs y paginas alternas.<br>-No demanda tiempo la capacitación a una persona por la limitación de actividades que permite realizar | Existe un alto nivel de documentación especializada en la web. |
| **Portabilidad** | -Está disponible para todos los sistemas operativos. | --Está disponible para todos los sistemas operativos. | -Está disponible para todos los sistemas operativos | |
| **Software/ Producto** | -Contiene foro, blog y buzón de sugerencias para la resolución de fallas e inquietudes.<br>-Es una herramienta con alto nivel de madurez (6 años en el mercado).<br>-Presenta actualizaciones cada 2 meses | -Contiene blog, preguntas frecuentes y buzón de sugerencias para la resolución de fallas e inquietudes.<br>-Es una herramienta con alto nivel de madurez (13 años en el mercado).<br>-Presenta actualizaciones mensualmente | -Es una herramienta con alto nivel de madurez (8 años en el mercado). | -Es una herramienta con alto nivel de madurez (8 años en el mercado). |
| **Especificidad** | -Permite la administración de los roles y grupos de la metodología Scrum, además de esto permite asignar actividades correspondientes por cada rol<br>-Permite crear el product *backlog* de manera sencilla y dependiendo el sprint activo<br>Tiene un área de ensayo en la cual se pueden proponer ideas por parte del cliente o de los integrantes del equipo para que después las apruebe o desapruebe el product *owner*<br>-Permite la priorización de las actividades del product *backlog*<br>-Permite desarrollar a través del tablero Kanban el sprint activo<br>-Permite registrar los datos necesarios para | -Permite la creación de roles<br>-Permite priorizar las actividades del product *backlog* de manera ágil<br>-Muestra un tablero en el cual se puede visualizar fechas y progreso de las actividades del product *backlog*<br>-Permite gestionar de manera visual las necesidades del product *backlog* de manera sencilla y constante en la ventana principal<br>-Permite crear el *done* para actividad<br>-Permite asignar el tiempo a cada actividad del *done*<br>-Muestra el diagrama de pert<br>-Permite registrar el progreso de cada actividad del product *backlog* | -Permite crear de manera ágil el product *backlog* en un sprint especifico<br>-Permite visualizar el burdonchart<br>-Permite visualizar de manera constante el product *backlog* a lo largo del desarrollo del proyecto | -Permite desarrollar de manera ágil y clara el sprint actual<br>-Permite registrar los datos de las métricas previamente preestablecidas de manera ágil y concreta<br>-Muestra gráficamente la información recopilada a partir de las métricas<br>-Permite definir el *done* de manera sencilla<br>-Permite visualizar de manera rápida el product backog dependiendo el sprint que se está desarrollando, por otra parte, permite ver las actividades del *done*<br>-Permite visualizar el tiempo faltante para completar una actividad del *done* y del product *backlog* |

| | Herramientas Evaluadas | | | |
|---|---|---|---|---|
| **Criterio** | **IceScrum** | **GanttProject** | **Scrumpy** | **Sprintometer** |
| | las métricas preestablecidas por la herramienta<br>-Permite crear el *done* dependiendo cada historia de usuario y verificar el flujo de trabajo de la misms<br>-Muestra gráficamente los datos de los diferentes sprints como las actividades realizadas, no finalizadas, tiempos que tardaron en realizarse, entre otras<br>-Tiene una sección para la visualización de del product *backlog*<br>-Permite asignarle a cada tarea del *done* un tiempo de desarrollo<br>-Permite la asignación de las diferentes actividades a cada integrante dependiendo las habilidades y sus roles asignados.<br>-Permite el trabajo colaborativo en la realización de las actividades por parte de todos los integrantes del equipo de manera simultánea<br>-Posee una línea de tiempo para cada sprint | | | |

Desventajas de herramientas evaluadas.

| | Herramientas Evaluadas | | | |
|---|---|---|---|---|
| **Criterio** | **IceScrum** | **GanttProject** | **Scrumpy** | **Sprintometer** |
| **Aceptación/ Usabilidad** | No permite el funcionamiento de la herramienta a través del teclado<br>-No posee ayudas de funcionalidad<br>-Se requiere conocimientos de configuración a través de la consola de los diferentes sistemas operativos | -No tiene mensajes de ayuda y error en todas las funcionalidades<br>-No permite el funcionamiento de la herramienta a través del teclado<br>-No contiene ayudas de funcionamiento | -Tiene limitado el trabajo de la herramienta a través del teclado -No contiene ayudas de funcionamiento<br>-La interfaz no es intuitiva ya que es compleja a la hora de trabajar tiene secciones difíciles de comprender<br>-No permite cambiar de idioma<br>-No permite cambiar la apariencia de la aplicación<br>-Se requiere conocimientos de | -Posee alertas de progreso y error en ciertas áreas de trabajo<br>-Tiene limitado el trabajo de la herramienta a través del teclado<br>-La distribución de la herramienta complica el funcionamiento de la misma, y por la falta de ayudas no se realizan las actividades de manera sencilla<br>-No se pueden relacionar todos los iconos y menús con su correspondiente funcionamiento |

| | Herramientas Evaluadas | | | |
|---|---|---|---|---|
| **Criterio** | **IceScrum** | **GanttProject** | **Scrumpy** | **Sprintometer** |
| | | | configuración para archivos de tipo .jar<br>-La instalación tarda aproximadamente 20 min | -No permite cambiar la apariencia de la aplicación |
| **Administración** | | -No tiene control de versiones completa, solo están las ultimas 2<br>-No permite ninguna clase de colaboración o interacción por parte del equipo | -No permite ninguna clase de colaboración o interacción por parte del equipo, ya que no permite la creación de diferentes usuarios | -No contiene una sección para el manejo y control de las versiones<br>-No permite ninguna clase de colaboración o interacción por parte del equipo, ya que no permite la creación de diferentes usuarios |
| **Eficiencia** | | -Solo se pueden crear usuarios en la maquina principal<br>-No permite crear grupos<br>-No hay opciones para permisos de usuarios<br>-No tiene opción de login | -No permite gestionar usuarios, grupos y permisos ya que no tiene ninguna opción para la creación de nuevos participantes<br>-No tiene opción de login | -No permite gestionar usuarios, grupos y permisos ya que no tiene ninguna opción para la creación de nuevos participantes<br>-Genera inconsistencia en los datos cuando el usuario se queda sin internet<br>-No es estable ya que presenta errores al momento de trabajar y guardar el progreso<br>-No tiene opción de login |
| **Entrenamiento** | -No existe documentación externa actualizada de la última versión | -No existe documentación externa, solo la que brinda la página oficial<br>-No existe documentación externa actualizada de la última versión a excepción de la brindada por la página oficial | -No existe documentación externa actualizada de la última versión a excepción de la brindada por la página oficial | -No existe documentación externa actualizada de la última versión a excepción de la brindada por la página oficial<br>-Demanda tiempo la capacitación de una persona debido a la complejidad de algunas características presentadas por la herramienta |
| **Integración** | No permite integrarse con ninguna herramienta de desarrollo | No permite integrarse con ninguna herramienta de desarrollo | No permite integrarse con ninguna herramienta de desarrollo | No permite integrarse con ninguna herramienta de desarrollo |
| **Portabilidad** | -Está disponible únicamente como plataforma web | -Está disponible únicamente para escritorio | -Está disponible únicamente para escritorio | -Solo está disponible para plataformas Windows<br>-Está disponible únicamente para escritorio |
| **Software/ Producto** | | | -Contiene únicamente foros y preguntas frecuentes para la resolución de fallos e inquietudes<br>-No presenta actualizaciones desde su última versión lanzada en 2009 | -Contiene buzón de sugerencias y preguntas frecuentes para la resolución de fallas e inquietudes<br>-No presenta actualizaciones desde su última versión lanzada en 2012 |

| | Herramientas Evaluadas | | | |
|---|---|---|---|---|
| **Criterio** | **IceScrum** | **GanttProject** | **Scrumpy** | **Sprintometer** |
| **Especificidad** | -No contiene ningún espacio para la planeación, desarrollo y evaluación del *planning meeting*<br>-No permite desarrollar el sprint review<br>-No contiene ningún espacio para el desarrollo de la retrospectiva<br>-No permite registrar métricas<br>--No contiene ningún espacio para la gestión del daily meeting<br>-No contiene espacios para motivar al equipo de trabajo<br>-No permite visualizar y gestionar el estado anímico del equipo de trabajo<br>-No contiene espacios para compartir conocimientos<br>-No contiene un espacio para la resolución de problemas presentaos en el proyecto | -No contiene un espacio en el cual se pueda identificar las necesidades por parte del cliente<br>-No permite crear diferentes sprints<br>-No contiene ningún espacio para la planeación, desarrollo y evaluación del *planning meeting*<br>-No permite desarrollar el sprint review<br>-No contiene ningún espacio para el desarrollo de la retrospectiva<br>-No permite trabajo colaborativo debido a que el proyecto solo está disponible para una maquina<br>-No contiene espacios para motivar al equipo de trabajo<br>-No permite visualizar y gestionar el estado anímico del equipo de trabajo<br>-No contiene espacios para compartir conocimientos<br>-No contiene un espacio para la resolución de problemas presentaos en el proyecto | -No permite la gestión de roles ni usuarios dentro del proyecto<br>-No contiene un espacio en el cual se pueda identificar las necesidades por parte del cliente<br>-No permite la priorización de las actividades del product *backlog*<br>-No contiene ningún espacio para la planeación, desarrollo y evaluación del *planning meeting*<br>-En el desarrollo del sprnt solo permite asignar fechas de finalización a las actividades<br>-No contiene ningún espacio para el desarrollo de la retrospectiva<br>-No permite registrar métricas<br>-No permite registrar información para las métricas establecidas<br>-No permite crear el *done* a las actividades del product *backlog*<br>-No permite comparar los diferentes sprints<br>-No contiene ningún espacio para la gestión del daily meeting<br>-No permite la creación de actividades de acuerdo al rol<br>-No permite trabajo colaborativo debido a que no se puede crear usuarios en el proyecto<br>-No contiene espacios para motivar al equipo de trabajo<br>-No permite visualizar y gestionar el estado anímico del equipo de trabajo<br>-No contiene espacios para compartir conocimientos<br>-No contiene un espacio para la resolución de problemas presentaos en el proyecto | -No permite la gestión de roles ni usuarios dentro del proyecto<br>-No contiene un espacio en el cual se pueda identificar las necesidades por parte del cliente<br>-No permite ordenar el product *backlog* por la prioridad de la actividad<br>-No contiene ningún espacio para la planeación, desarrollo y evaluación del *planning meeting*<br>-No permite desarrollar el sprint review<br>-No contiene ningún espacio para el desarrollo de la retrospectiva<br>-No permite registrar métricas<br>-No permite comparar los diferentes sprints<br>-No contiene ningún espacio para la gestión del daily meeting<br>-Permite definir el tiempo para las actividades del *done* en días<br>-No permite la creación de actividades de acuerdo al rol<br>-No permite trabajo colaborativo debido a que no se puede crear usuarios en el proyecto<br>-No contiene espacios para motivar al equipo de trabajo<br>-No permite visualizar y gestionar el estado anímico del equipo de trabajo<br>-No contiene espacios para compartir conocimientos<br>-No contiene un espacio para la resolución de problemas presentaos en el proyecto |