Propuesta Basada en Scrum, Peopleware y Software Libre: Caso Universidad de Nariño[1]

Geovany Steven Viteri Salazar, Magister en Software Libre (Universidad Autónoma de Bucaramanga), Ingeniero de Sistemas (Universidad de Nariño); Ingeniero Líder de Proyecto (Universidad de Nariño); Correo electrónico personal: gevisa82@hotmail.com

Giovanni Albeiro Hernández Pantoja, Magister en Docencia Universitaria (Universidad de Nariño), Especialista en Gerencia Informática (Corporación Universitaria Remington), Ingeniero de Sistemas (Universidad de Nariño); Profesor Asociado Universidad Mariana (Nariño, Colombia), integrante del grupo de investigación GISMAR; Correo electrónico institucional: gihernandez@umariana.edu.co

Resumen

Este trabajo presenta la consolidación de una metodología para el soporte, mantenimiento y construcción de software; basada en Scrum, Peopleware y apoyada por herramientas de software libre, en el Centro de Informática (CI) de la Universidad de Nariño. Está investigación, fue de corte cuantitativo, con un enfoque empírico-analítico de tipo descriptivo y propositivo. La población objeto de estudio es la Universidad de Nariño, específicamente el CI. Los principales resultados fueron la comparación de los elementos metodológicos del proceso de software del CI y Scrum. Con base en la comparación, se logra plantear una propuesta basada en Scrum y Peopleware, soportada por Kunagi. El trabajo permite concluir que en el CI, no se evidencia un método que propicie el trabajo en equipo. Las principales similitudes entre el proceso de software del CI y Scrum, son: pila de producto, done y development team. La principal diferencia es la ausencia de métricas.

Palabras clave: Construcción de Software, Peopleware, Scrum.

Proposal Based on Scrum, Peopleware and Free Software: Case Nariño University

Abstract

[1] Este artículo es el resultado de la investigación titulada: metodología adaptativa basada en Scrum, apoyada por herramientas de software libre para el centro de informática de la Universidad de Nariño, desarrollada desde el 17 de junio de 2016 hasta el 11 de agosto de 2017 en San Juan de Pasto, departamento de Nariño, Colombia.

This work presents the consolidation of a methodology for the support, maintenance and construction of software; based on Scrum, Peopleware and supported by free software tools, in the Computer Center (CI) of the University of Nariño. This research was quantitative, with an empirical-analytical approach of a descriptive and propose type. The study population is the Nariño University, specifically the CI. The main results were the comparison between the methodological elements of the software process of CI and Scrum. From on the comparison, it was possible to do a proposal based on Scrum and Peopleware, supported by Kunagi. The work allows to conclude that in the CI, there isn't evidence of a method that encourages teamwork. The main similarities between the software process of CI and Scrum are: product backlog, done and development team. The main difference is the absence of metrics.

Key words: Peopleware, Scrum, Software Development.

Resumo

Este trabalho apresenta a consolidação de uma metodologia para o suporte, manutenção e construção de software; based en Scrum, Peopleware e apoiados por ferramentas de software livre, no Centro de Informática (CI) da Universidade de Nariño. Estudo, foi de corte cuantitativo, com um enfoque empírico-analítico de tipo descritivo e propositivo. A população objeto de estudo é a Universidad de Nariño, especificamente o CI. Principais resultados foram as comparação dos elementos metodológicos do processo de software do CI e do Scrum. Con base em comparação, se logra plantear uma solução em Scrum y Peopleware, soportada por Kunagi. O trabalho permite concluir que na CI, não é uma evidencia um método que propicie o trabalho em equipo. As principais semelhanças entre o processo de software do CI e Scrum, filho: pila de produto, feito e equipe de desenvolvimento. A principal diferença é a ausencia de métricas.

Introducción

Las empresas, en cuanto a operación o funcionamiento, han venido avanzando en los diferentes factores que intervienen al momento de construir software, aspecto relevante para determinar el éxito o fracaso de los proyectos. Según el Stadish Group (2016), en el informe que presenta en el chaos manifiesto, para el año 2015 del total de proyectos de construcción de software evaluados para identificar el porcentaje de éxito, el 29% finalizan con éxito, un 52% son catalogados como cuestionables, ya que tuvieron desfases en el tiempo, presupuesto, en las características y funcionamiento del software, o en alguna combinación de las anteriores; y un 19% fracasaron. En este informe se puede apreciar que, se ha mantenido estable la tasa de éxito de los proyectos en los últimos años. El panorama para el país y la región, está en relación con los datos del chaos manifiesto y pueden ser más desalentadores.

Por otra parte, la construcción de software es una actividad compleja, que actualmente, se realiza por equipos, que deben dar respuesta a las necesidades crecientes de software en las organizaciones. No obstante, el trabajar en equipo tiene un conjunto de factores que inciden al momento de alcanzar los objetivos, que esta labor traza. Putman, citado por McConell (McConnell, 2006) logra demostrar que cuando el tamaño del equipo crece por encima de un grupo de personas, el esfuerzo se aumenta, pero el tiempo del proyecto, no se reduce. DeMarco (DeMarco, 1999), plantea que un equipo con exceso de trabajo, ocupado y sobresaturado no es garantía de mayor efectividad; y además, no permite visualizar mayores beneficios para un proyecto. Así mismo, Brooks (Brooks, 1995) a partir de la experiencia, éxito y fracaso en el desarrollo de software, logra concluir que, en un equipo de desarrollo, agregar más personas a un proyecto que tiene un retraso, provocará un retraso mayor. En trabajos intelectuales, como programar, escribir artículos, entre otros, las interrupciones son un mal que afecta enormemente a la productividad. Parnin y Rugaber (Parnin & Rugaber, 2010) hicieron un estudio en el año 2010 sobre las interrupciones, siendo la conclusión más destacada que, lo normal es que a un programador le lleve de 10 a 15 minutos volver al estado de concentración previo al haber sido interrumpido. Otro aspecto fundamental, que incide en la productividad y desempeño de un equipo de trabajo es la comunicación. Los estudios realizados por Cockburn (Cockburn, 2015), permiten establecer que la forma más efectiva de comunicación corresponde a mantener al equipo interactuando en un sitio, frente a un tablero, formulando y respondiendo cuestionamientos e interrogantes.

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 el 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 la 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 tienen 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. De igual manera, 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 los cambios continuos que se hace al software y la falta de actualización de los mismos. 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 de 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).

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 del costo de operación

Para el desarrollo de la investigación, se encontraron trabajos investigativos orientados a describir la experiencia de haber utilizado Scrum en la construcción de un producto software (Bannerman, Hossain, & Jeffery, 2012), (May, Morales, Marrufo, & Martín, 2013). Estas investigaciones permitieron identificar aciertos y dificultades en la aplicación de los principios definidos en Scrum en la elaboración de software. Otro espacio investigativo explorado, fue la revisión de trabajos que evaluaron las prácticas de trabajar con Scrum en empresas de la Industria de Software (Hernández, Martínez, Argote, & Coral, 2015), (De Souza, Zambalde, Tonelli, Souza, Zuppo, & Rosa, 2014), (Martínez, Ramón, & Bertone, 2012), (Colla, 2012) y (Vlaanderen, Jansen, Brinkkemper, & Erik, 2011). Estos estudios permitieron identificar aciertos y dificultades en la aplicación de los principios definidos en Scrum en empresas de la Industria de Software.

Los antecedentes consultados lograron mostrar un camino investigativo con aportes para las organizaciones que desean adoptar Scrum como parte de la forma de trabajo, definir el soporte teórico de la propuesta y logran mostrar una brecha investigativa, debido a que ninguno de los estudios revisados aborda Peopleware como un elemento para intervenir la complejidad de trabajar en equipo construyendo software.

La Construcción de Software hace referencia a 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, lo cual depende de sus características y alcance. En este sentido, una metodología para la Construcción de Software es el conjunto de técnicas, procedimientos, métodos, herramientas y soporte documentado para el diseño y desarrollo de software; además debe definir con precisión roles y actividades, así mismo prácticas y técnicas para adaptarlas al proyecto.

De acuerdo con Shwaber y Sutherland (2013), Scrum es un marco de trabajo mediante el cual las personas pueden intervenir problemas complejos adaptativos, a la vez que entregan

productos del máximo valor posible en forma productiva y creativa. Scrum, se enfoca en agregar valor a los procesos de negocio de los clientes mediante la verificación continua, adaptación e innovación.

Para DeMarco y Lister (1999) peopleware es un enfoque que plantea como realizar la gestión de la complejidad del trabajo en equipo, cuando se construye software, estableciendo estrategias para lograr equipos eficientes y efectivos.

Teniendo en cuenta el camino teórico que fundamenta la construcción de software utilizando una metodología ágil como Scrum y la gestión de la complejidad del trabajo en equipo, se planteó como propósito principal, en el trabajo de investigación que soporta este artículo, consolidar una metodología para el soporte, mantenimiento y construcción de software; basada en Scrum, Peopleware y apoyada por herramientas de software libre, en el Centro de Informática - CI de la Universidad de Nariño. Para alcanzar este fin, en una primera fase se caracterizó 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. Posteriormente, se realizó un análisis comparativo los elementos metodológicos del proceso de construcción de software del Centro de Informática de la Universidad de Nariño y los de Scrum. Finalmente, se formuló una propuesta de trabajo basada en Scrum, Peopleware y apoyada por herramientas de software libre para el Centro de Informática de la Universidad de Nariño.

Este trabajo investigativo es interesante porque, al caracterizar los elementos metodológicos del proceso de construcción de software del CI de la Universidad de Nariño y los definidos en el marco de trabajo Scrum se conoce y analiza 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.

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 logra 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 es descrito en una propuesta basada en este marco de trabajo, como 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, 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 puede contar con indicadores para la toma de decisiones por parte del director, las cuales ya no se basarán únicamente en su experiencia y percepción.

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, incorporando técnicas de peopleware 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.

Este documento comienza con la descripción de la metodología, aquí se explica la forma como se desarrolló la investigación. Posteriormente, se muestran los resultados obtenidos y se hace una discusión acerca de algunas consideraciones y reflexiones frente a los hallazgos y finalmente se presentan las conclusiones.

Desarrollo

Metodología

Para cumplir con el primer objetivo se tuvo como fuente primaria de información, a los funcionarios del CI de la Universidad de Nariño. Para recopilar los datos, se aplicó una encuesta. Los datos recolectados, se analizaron mediante estadística descriptiva. Los instrumentos de recolección de información en todo el proyecto fueron validados con la técnica de juicio de expertos. Como resultado se obtuvo un documento con la caracterización de los elementos metodológicos del CI.

El segundo objetivo tomó como fuente primaria de información, el documento con la caracterización elaborado en el primer objetivo y los referentes teóricos de Scrum, a partir de los cuales, se realizó una análisis documental. El resultado alcanzado fue, una matriz de comparación de los elementos metodológicos del proceso de construcción de software del CI con los de Scrum.

El tercer objetivo tomó como fuente primaria de información, la matriz construida en el segundo objetivo y los referentes teóricos de Peopleware, a partir de los cuales, se realizó una análisis documental y se elabora una propuesta metodológica. El resultado alcanzado fue, un documento con las adopciones de la propuesta.

Resultados

Los resultados que a continuación se presentan, describen inicialmente a la población de informantes que participaron en la validación de la propuesta metodológica y se detalla las percepciones de los informantes de acuerdo con las variables.
La población encuestada estuvo conformada por 7 funcionarios del Centro de Informática de la Universidad de Nariño.

Como se puede observar en la Tabla 1, el 71.4% de los funcionarios encuestados pertenecen al género masculino y el 28.6% al género femenino.

Tabla 1
Distribución de funcionarios por género

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

Fuente: La presente investigación

En la tabla 2, 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 2
Distribución de funcionarios por edad

Rango	FO - Frecuencia Observada	Frecuencia Observada (%)
26 y 30	3	42,9
31 y 35	1	14,3
> 35	3	42,9
Total	7	100

Fuente: La presente investigación

En la tabla 3, se indica el tiempo (años) que los funcionarios han trabajado en el CI, se puede observar que la rotación del personal no es alta, debido a que la mayoría han venido 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.

Tabla 3
Distribución de funcionarios por tiempo de trabajo.

Rango (años)	FO - Frecuencia Observada	Frecuencia Observada (%)
Menos de 1 año	1	14,2
Entre 1 y 2 años	2	28,6
Entre 2 y 3 años	1	14,2
Entre 3 y 5 años	1	14,2
Más de 5 años	2	28,6
Total	7	100

Fuente: La presente investigación

En cuanto al cargo que desempeña cada uno de los funcionarios, no existe unicidad y claridad en la denominación, probablemente obedezca a las múltiples funciones que cumple cada empleado impidiéndole reconocer la actividad principal que realiza.

En la tabla 4, se puede observar el tiempo que los funcionarios llevan desempeñando el cargo, es el mismo que llevan trabajando en la dependencia, lo que permite identificar que no han existido cambios en la organización del equipo de trabajo.

Tabla 4
Distribución de funcionarios por tiempo de desempeño en el cargo

Rango (años)	FO - Frecuencia Observada	Frecuencia Observada (%)
Menos de 1 año	1	14,2
Entre 1 y 2 años	2	28,6
Entre 2 y 3 años	1	14,2
Entre 3 y 5 años	1	14,2
Más de 5 años	2	28,6
Total	7	100

Fuente: La presente investigación

El CI cuenta con un 57,1% de funcionarios con nivel de formación profesional y un 42,9% con nivel de especialización. Esta información permite establecer el grado de preparación con 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.

El análisis de la información recopilada en relación con los indicadores: etapa, actividad, rol, artefacto, herramienta y lineamiento (Ver figura 1), permitió establecer 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 la codificación y una vez terminado el producto integrarlo al sistema en producción muchas veces sin 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. Al no realizar una 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.

Por otra parte, 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.

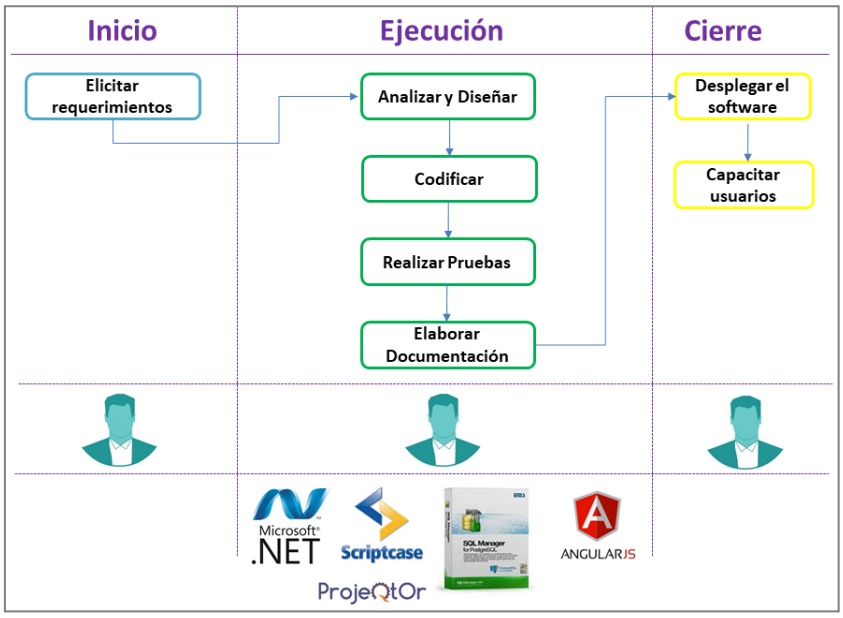

Figura 1. Síntesis del proceso de soporte, mantenimiento y construcción de software. Elaborada en esta investigación.

Al realizar el análisis comparativo de los elementos metodológicos del Centro de Informática y el marco de trabajo Scrum (Ver Figura 2), 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 cuando 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 y frecuentes del producto, por el contrario al finalizar el desarrollo, se aplican pruebas, se despliega la solución y se presenta al cliente/usuario. Tampoco se hace retroalimentación en el proceso o lo que se conoce en Scrum como revisión y retrospectiva, cuyo objetivo es hacer un autoanálisis de cómo se está trabajando identificando fortalezas y debilidades para tomar correctivos a tiempo.

Para el indicador 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, tareas, elaboración de artefactos como el sprint backlog y la estimación de las necesidades.

Para el indicador 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.

En el indicador artefacto, la semejanza más clara con Scrum es la del Product Backlog, con la lista de requerimientos que el funcionario del CI obtiene a través de reuniones con los interesados. En lo relacionado con métricas, no se evidencia una definición explicita y especificada, que permita evaluar el proceso de creación, soporte y mantenimiento del software.

En cuanto al indicador lineamiento, no hay similitudes porque no se recopilan datos, ni se definen métricas que permitan evaluar el desarrollo, soporte y mantenimiento del software a cargo de la dependencia.

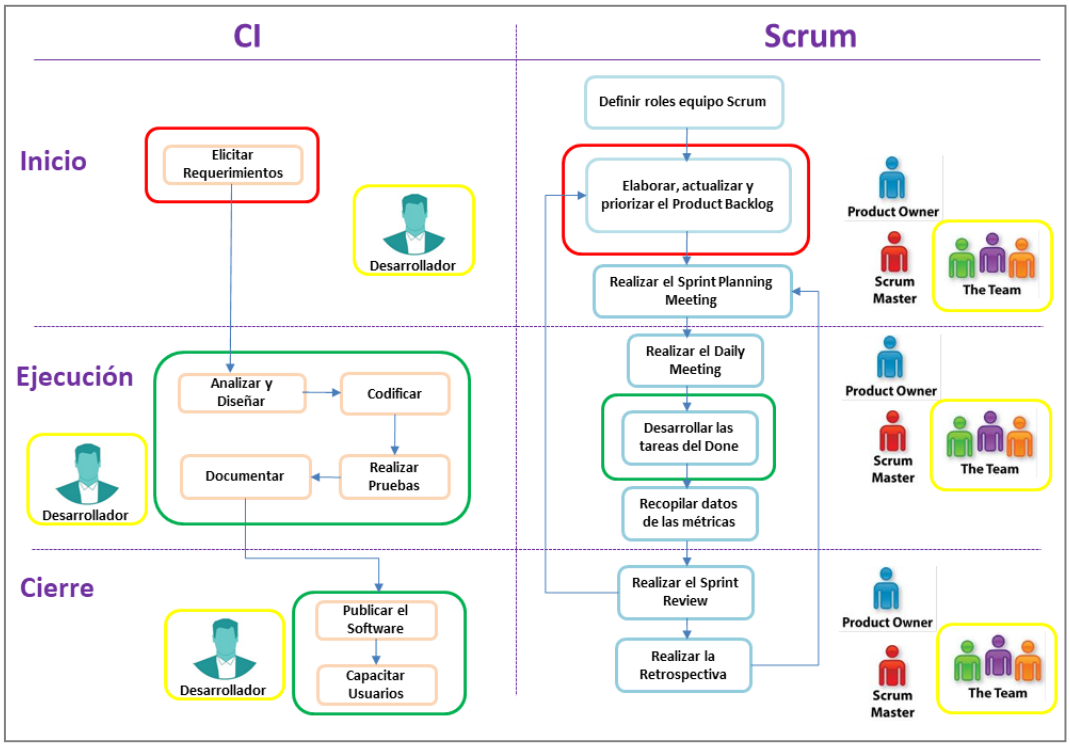

Figura 2. Contraste proceso CI y Scrum. Elaborada en esta investigación.

Para determinar que herramienta de software libre utilizar como soporte a la gestión de Scrum, inicialmente se realizó la evaluación de Kunagi, Scrumpy, Sprintometer, Icescrum y Ganttproject; utilizando OAM-F/OSS (open appraisal model for free and open source software) planteado por Jiménez (2016), 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.

En la tabla 5, se puede observar las puntuaciones obtenidas por cada una de las herramientas en cada criterio de evaluación (TC) con el correspondiente ponderado (VF).

Tabla 5
Valoración Final de la evaluación de las herramientas de software libre

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	12.0	35	14.0
Administración	10	15	10.0	7	4.7	3	2.0	15	10.0	3	2.0
Eficiencia	10	28	9.3	14	4.7	15	5.0	30	10.0	18	6.0
Entrenamiento	10	13	8.7	10	6.7	8	5.3	12	8.0	8	5.3
Integración	10	1	2.0	1	2.0	1	2.0	1	2.0	1	2.0
Portabilidad	10	7	7.0	6	6.0	2	2.0	6	6.0	6	6.0
Software/Producto	5	17	3.4	12	2.4	16	3.2	21	4.2	10	2.0
Especificidad	25	91	19.0	35	7.3	48	10.0	72	15.0	42	8.8
Total (%)			73		42.9		42.7		67.2		46.1

Fuente: La presente investigación

Como se puede apreciar en la valoración total de la Tabla 5, la herramienta Kunagi es la alternativa más recomendable, ya que obtuvo un ponderado final de 73%.

Para el criterio *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 deben 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 documentación en las diferentes versiones, en lo relacionado a la interacción de los stakeholders la herramienta cuenta con wikis, blogs, foros 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 privilegios de acuerdo con las directrices de la metodología Scrum, permitiendo el acceso únicamente a las actividades o elementos indicados para este usuario. Además, cuenta con una base de datos en la cual almacena la información sobre los proyectos y sus especificidades, realizando backups de forma automática, lo cual permite restaurar los datos ante una posible falla o error.

Evaluando el criterio *Entrenamiento*, en lo relacionado con la documentación, Kunagi cuenta con información actualizada en la 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, el manejo es fácil e intuitivo lo que disminuye 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 el sitio web con un formulario donde se pueden realizar consultas y también visualizar el historial de las mismas realizadas por otros usuarios. A pesar de que es una herramienta que cuenta con un grado de madurez de 6 años, una desventaja es que las actualizaciones son casi anualmente, dificultando la evolución del producto.

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 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 de gestionar, 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. Además el uso de menús contextuales facilita la identificación de las actividades a realizar sobre un elemento seleccionado.

Algo muy interesante de Kunagi es la definición del Done, 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.

Otro aspecto relevante es que la herramienta permite al líder del proyecto gestionar la asignación de las actividades a cada integrante y seleccionar el estado de ánimo con el cual se desarrolla la actividad. También permite la creación de blogs, wikis y chats para el equipo con el fin de compartir conocimiento o para resolver problemas presentados.

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 los objetivos del sprint.

Aunque como se describió anteriormente las ventajas que posee esta herramienta son muchas, de la misma manera hay algunas desventajas como la falta de una funcionalidad donde se pueda desarrollar el planning meeting y registrar otras métricas diferentes a las que trae por defecto. Además no permite realizar una comparación entre sprints.

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 las necesidades planteadas por los expertos al momento de crear software.

La técnica “Niko Niko[2]” también conocida como calendario de la felicidad o índice de la felicidad, consiste en organizar un calendario en el cual cada fila corresponde 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.

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

Después de ciertos periodos de tiempo por ejemplo al realizar la revisión o la retrospectiva del sprint, se inspecciona 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[3]" 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 las interrupciones y a propiciar eficiencia en el trabajo.

La técnica "Mob Programming[4]", 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 donde uno de los participantes "piloto" es el encargado de realizar la acción compleja, por ejemplo diseñar o codificar, mientras 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 "Coding Dojo[5]" 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 CI es la dificultad que generaría el cambio en la forma como vienen trabajando los funcionarios, saliendo de la 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.

En una primera etapa, como se muestra en la Figura 3, 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 instala y

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

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

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

configurará Kunagi[6], 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, se debe realizar 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 sprint. En este primer sprint, es recomendable definir las tareas del Done (hecho) que permiten tener claro que se debe hacer para poner en funcionamiento en el cliente un requerimiento o historia de usuario. Las tareas seleccionadas serán estrictamente aquellas que agreguen valor al producto software y su definición estará a cargo del *Development Team*.

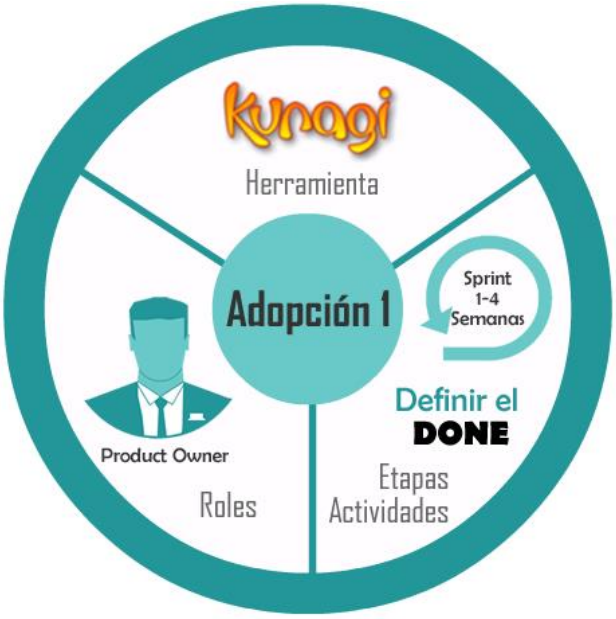

Figura 3. Adopción primera de la propuesta. Elaborada en esta investigación.

En una segunda etapa, como se muestra en la Figura 4, 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 con 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 a los 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 responsables, en donde todos los integrantes del *Development Team* podrán observar el estado de las mismas de manera gráfica a través del tablero kanban [7] que

[6] Para más información acerca de la herramienta visitar http://kunagi.org/
[7] Para más información sobre el tablero kanban visitar https://es.atlassian.com/agile/kanban

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*.

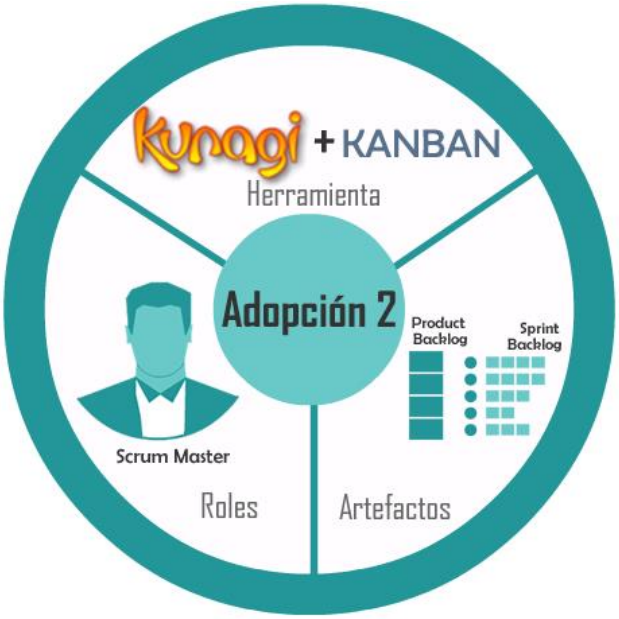

Figura 4. Adopción segunda de la propuesta. Elaborada en esta investigación.

Para la tercera etapa, como se muestra en la Figura 5, 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 debe desarrollar la capacidad de autogestionarse y ser 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 el proceso de negocio del cliente. En este sentido, se utiliza la técnica "Niko Niko", la cual permite registrar el estado de ánimo de cada integrante, antes de iniciar una jornada laboral; esta información es valiosa para el Scrum Master al momento de buscar un equipo motivado y realizar 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 es fomentar una buena práctica en el Development Team.

En la tercera 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 la visualización del trabajo en la herramienta Kunagi, a través del Burndown Chart. Este artefacto es un reporte gráfico que muestra el trabajo que falta por realizarse para lograr el objetivo definido para el sprint y visualizar el estado del proyecto.

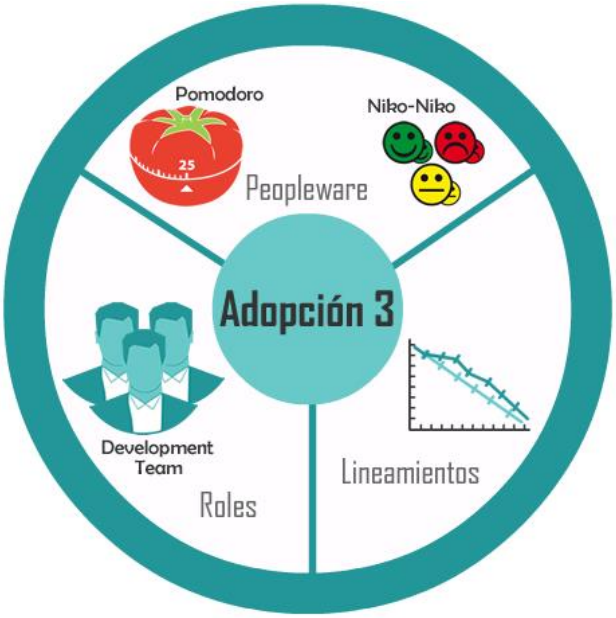

Figura 5. Adopción tercera de la propuesta. Elaborada en esta investigación.

Para la cuarta adopción, como se muestra en la Figura 6, se sugiere continuar con el empoderamiento del *Development Team*, aportando elementos para que los miembros del equipo tengan la independencia de organizar y gestionar el trabajo, respetando las opiniones y aportes de todos, propiciando la eficiencia y efectividad. Para ello se utiliza la técnica "Mob programming", en la cual el equipo trabaja sobre una tarea o actividad concreta y compleja al mismo tiempo, en un mismo espacio y tiempo.

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 programan espacios de reunión de todo el *Development Team*, donde se desarrollan tareas propias del proyecto o aspecto nuevos por aprender, fomentando de esta manera la construcción colectiva del conocimiento. En esta fase, 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.

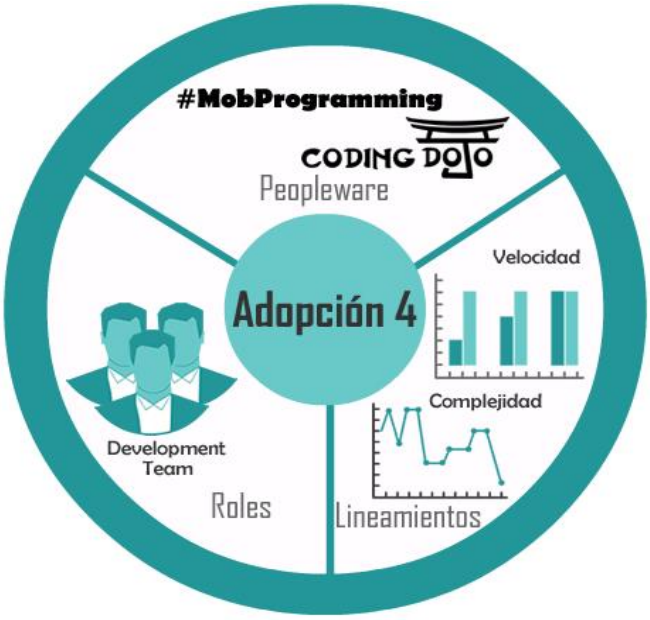

Figura 6. Adopción cuarta de la propuesta. Elaborada en esta investigación.

Discusión

El desarrollo de este trabajo investigativo tiene como propósito especial implícito, proponer formas de intervenir, no únicamente organizaciones públicas de Educación Superior, sino por el contrario, ser la punta de lanza para llegar a cualquier entidad que comprenda que la tecnología es un aliado primordial al momento de cumplir con los objetivos estratégicos. El software como un producto tecnológico esencial en este camino, debe tener la relevancia que le corresponde, sin olvidar que hacerlo, soportarlo y mantenerlo es una tarea compleja; más aún cuando se ha logrado demostrar desde trabajos como los de Cockburn (2015), Putnam (2012), McConell (2006), DeMarco (2001) y Brooks (1995) que los aspectos relacionados con las personas son lo que tienen mayor impacto en la eficiencia, productividad y calidad al momento de construir, soportar o mantener software.

Los antecedentes encontrados permitieron visualizar un interés por describir experiencias en la apropiación de Scrum, tanto a nivel académico, como en la Industria de Software. No obstante, un equipo requiere desplegar y perfeccionar un conjunto de competencias para ser productivo cuando hace software. Zaraket, Olleik, y Yassine (2014, pág. 310) clasifican estas competencias en tres (3) categorias: técnicas, comunicativas y para la gestión de equipo. Para poder consolidar un equipo de trabajo efectivo y eficiente, no es suficiente con haber desarrollado competencias unicamente en lo técnico o apropiar un conjunto de lineamientos que propicien el trabajo en equipo. En este orden de ideas, cobra relevancia académica, investigativa e industrial, analizar elementos técnicos como los definidos en el área de Peopleware, que posibiliten consilidar equipos donde se propicie el desarrollo de habilidades comunicativas y de gestión del talento humano.

El camino investigativo recorrido permite confirmar que no existen fórmulas mágicas para consolidar equipos efectivos y eficientes, por el contrario; se hace necesario partir del quehacer de las organizaciones que hacen software, para plantear alternativas propias que enriquezcan y agreguen valor al trabajo en equipo. Los investigadores por el momento creemos, que las adopciones puras de propuestas teóricas para el trabajo en equipo, no necesariamente dejan prever acciones exitosas, por el contrario, generan crisis y caos.

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 un conjunto de herramientas de software libre para soportar una propuesta metodológica basada en Scrum y Peopleware, 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 propuestos por el modelo OAM-F/OSS.

Un aspecto a tener en cuenta en la construcción 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.

Este trabajo investigativo presenta una propuesta para trabajo en equipo basada en Scrum y técnicas de Peopleware 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.

Referencias Bibliográficas

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Erik, J. (2011). The agile requirements refinery: Applying SCRUM principles to software product management. *Information and Software Technology*, 58-70.

Bannerman, P., Hossain, E., & Jeffery, R. (2012). Scrum Practice Mitigation of Global Software Development Coordination Challenges: A Distinctive Advantage? *45th Hawaii International Conference on System Sciences* (págs. 5309 - 5318). Hawaii: IEEE Computer Society.

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., y otros. (2001). *Manifiesto por el Desarrollo Ágil de Software.* Recuperado el 22 de Mayo de 2014, de http://agilemanifesto.org/iso/es/manifesto.html

Brooks, F. (1995). *The Mythical Man-Month: Essays in Software Engineering.* Addison-Wesley.

Castillo, E. J. (14 de Junio de 2016). Entrevista al director del Centro de Informática de la Universidad de Nariño. (S. V. Salazar, Entrevistador) San Juan de Pasto - Colombia.

Cockburn, A. (23 de 11 de 2015). *Effectiveness of different modes of communication graph*. Recuperado el 17 de 1 de 2017, de http://alistair.cockburn.us/Effectiveness+of+different+modes+of+communication+graph.gif

Colla, P. (2012). Marco para evaluar el valor en metodología SCRUM. *13th Argentine Symposium on Software Engineering*, (págs. 32-46). la Plata - Argentina.

De Souza, P., Zambalde, A., Tonelli, A., Souza, S., Zuppo, L., & Rosa, P. (2014). Agile principles and achievement of success in software development: A quantitative study in Brazilian organizations. *Procedia Technology*, 718-727.

DeMarco, T., & Lister, T. (1999). *Peopleware : productive projects and teams.* New York - USA: Dorset House Publishing Co.

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. *28*(5).

Heshusius Rodríguez, K. (2009). Colombia: Desafíos de una Industria en Formación. En P. Bastos Tigre, & F. Silveira Marques, *Desafíos y oportunidades de la Industria de Software en América Latina.* Colombia: Mayol Ediciones S.A.

Jiménez, F. E. (2016). *MODELO DE EVALUACIÓN PARA LA SELECCIÓN DE HERRAMIENTAS DE SOFTWARE LIBRE EN EL PROCESO DE GESTIÓN DOCUMENTAL.* Bucaramanga: Universidad Autónoma de Bucaramanga.

Leber, J. (2014). *Fastcompany*. Recuperado el 27 de 05 de 2016, de https://www.fastcompany.com: https://www.fastcompany.com/3028160/happy-workers-are-more-productive-science-proves-it

Martínez, N., Ramón, H., & Bertone, R. (2012). Aplicabilidad de COMPETISOFT a partir de un método ágil como Scrum. *XVIII Congreso Argentino de Ciencias de la*

Computación (págs. 1-11). Buenos Aires: Red de Universidades con Carreras en Informática (RedUNCI).

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. *Congreso Interdisciplinario de Cuerpos Académicos, vol 2* (págs. 176-190). Merida: Universidad Tecnológica del Suroeste de Guanajuato.

McConnell, S. (2006). *software estimation demystifying the black art.* Redmond, Washington: Microsoft Press.

Parnin, C., & Rugaber, S. (2010). Resumption strategies for interrupted programming. *19*(1).

PROEXPORT COLOMBIA. (2015). *Inversión en el Sector Software y Servicios de TI en Colombia*. (PROCOLOMBIA, Exportaciones, Turismo, Inversión marca País.) Recuperado el 28 de Septiembre de 2015, de http://www.inviertaencolombia.com.co/sectores/servicios/software-y-servicios-de-ti.html

Quintero, J. B., & Anaya, R. (2007). MDA y el papel de los modelos en el proceso de desarrollo de software. *Escuela de Ingeniería de Antioquia*(8), 131-146.

Rojas Izaquita, M. E. (2011). *Agilizando lo ágil: un framework para el desarrollo de software bajo el modelo CMMI en compañias que utilizan metodologías ágiles de desarrollo de software usando el modelo acelerado de implmentación (AIM).* Bogotá (Colombia): Universidad Nacional de Colombia.

Schwaber, K. (2004). *Agile Project Management with Scrum.* Estados Unidos: Microsoft Press.

Schwaber, K., & Sutherland, J. (2016). *The Scrum guide*. Recuperado el 15 de Mayo de 2016, de http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100

The Standish Group. (2016). *CHAOS MANIFIESTO, Think big, Act Small.*