Facultad 5 – Laboratorio de Investigación en Gestión de Proyectos

Facultad 1 – Centro de Software Libre

Metodología para desarrollar la distribución cubana de GNU/Linux Nova.

Trabajo final presentado en opción al título de Máster en Gestión de Proyectos Informáticos

Autora:

Ing. Yusleydi Fernández del Monte

Tutores:

Dr.C. Febe Angel Ciudad Ricardo

Ms.C. Nilet Ma. Soto López

La Habana, Mayo de 2013

DEDICATORIA

Dedico esta tesis a mi familia, por ser lo más importante para mi,

por ser luz e iluminar mi vida.

En especial a mi mamita querida porque cada logro mío por derecho le pertenece,

todo lo que soy se lo debo a ella.

A la memoria de mi abuela Mima por los lindos recuerdos que me quedan, y porque sin ti yo no hubiera sido la persona que soy hoy, gracias por tus valores y dignidad.

AGRADECIMIENTOS

Agradezco a todos los que me han ayudado en la realización de esta investigación, que sin dudas cierra una parte importante en mi carrera profesional, de igual modo abre nuevos retos y horizontes.

Agradezco en especial,

A Dios, por tanta fe, amor, luz y bondad.

A mis padres porque cada uno me ha enseñado una manera diferente de ver la vida y sin darse cuenta han trazado un camino en mi, que tiene por meta constante ser mejor persona.

A mis abuelas por el amor tan lindo, por cada caricia y cada mirada brillante. A mi abuelo Félix por su cariño.

A mi hermanito, por sus lindas cualidades como persona y por regalarme tan bellas sobrinas.

A mi primito Arieski por hacerme reír con sus ocurrencias.

A mis tías y tíos por llenarme tanto de cariño, de tantas cosas que aprender y por sus consejos, en especial a mi tía Elena porque en esta etapa de mi vida es como una segunda madre para mi.

A mis primos y primas por los momentos lindos vividos.

A Olguere, Albertini y Aliam por ser más que primos, por ser mis amigos, mis confidentes, por malcriarme y exigirme siempre lo mejor de mi.

A Jesús por apoyarme siempre.

A mi Cukito por ser tan importante en esta etapa de mi vida, por su apoyo y su cariño.

A mis amigas y amigos, por cada risa, por cada consejo, por cada alegría y por cada charla interminable.

A mi eterno amigo Joe porque juntos diseñamos muchos sueños y este es uno de ellos.

A Miriam Cárdenas por ser un ejemplo a seguir como ser humano y profesional.

A mis amigos del Departamento Sistemas Operativos, a los que están, y a los que ya no están porque este trabajo es el resultado del esfuerzo de todos, de las ganas de ser más exitosos en lo que realmente hacemos muy bien.

A mis compañeros de trabajo, por su apoyo, por cada ayuda.

A mis tutores por apoyarme cuando los necesité, por ser tan excelentes guías en el logro de estos resultados.

A la dirección de la Maestría Gestión de Proyectos Informáticos por su apoyo para la culminación de estos estudios.

A mi profesor Tomás López Jimenez porque juntos diseñamos un plan de vida profesional y me llena de orgullo aun en su ausencia cumplir con las metas establecidas.

DECLARACIÓN JURADA DE AUTORÍA

Declaro por este medio que yo, Yusleydi Fernández del Monte, con carné de identidad 85013109854, soy el autor principal del trabajo final de maestría Metodología para el desarrollo de distribuciones GNU/Linux, desarrollada como parte de la Maestría en Gestión de Proyectos Informáticos y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de La Habana a los ___ días del mes de _____ del año _____.

RESUMEN

Uno de los protagonistas para lograr la migración a software libre y código abierto es la distribución cubana de GNU/Linux Nova. En el año 2010 se realizó una auditoría en los proyectos Nova Escritorio, Nova Ligero, Nova Base y Nova Servidores donde se detectaron un conjunto de desviaciones relacionadas con las áreas de gestión de la configuración de software, administración de requisitos, aseguramiento de la calidad y la planificación de los proyectos. Para dar solución a los problemas identificados se plantea como objetivo de la investigación diseñar una metodología de desarrollo de software que contribuya a la disminución de las desviaciones en el proceso de desarrollo de la distribución cubana GNU/Linux Nova. Como resultado de la investigación se obtuvo la metodología tradicional Nova - OpenUp compuesta por principios, prácticas, fases, disciplinas, actividades, objetivos, descripciones, roles, artefactos, buenas prácticas y sus relaciones. Para validar la metodología se utilizó el Modelo de evaluación de metodologías, los métodos Juicio de Expertos y Experimental determinando que es correcta y completa. La metodología ha sido puesta en práctica en los proyectos Nova Escritorio, Nova Ligero, Nova Base y Nova Servidores ayudando a eliminar desviaciones detectadas. Además, si se implementa adecuadamente sirve de apoyo a los proyectos que deseen alcanzar el nivel 2 de CMMI.

Palabras claves: Metodología, distribuciones de GNU/Linux, Nova.

ABSTRACT

One of the protagonists to achieve migration to open source and free software is the Cuban distribution of GNU / Linux Nova. In 2010 an audit in projects Nova Desk, Nova Light, Nova Base and Nova Servers was performed which detected a set of deviations related to the areas of configuration management, requirements management, quality assurance and project planning. To solve the problems identified therefore seeks research design a software development methodology that contributes to the decrease of the deviations in the development process of the Cuban distribution GNU / Linux Nova. As a result of the investigation was obtained traditional methodology Nova - OpenUp composed of principles, practices, phases, disciplines, activities, objectives, descriptions, roles, artifacts, best practices and relationships. To validate the methodology was used Model assessment methodologies, methods Experimental and Expert Judgment determining that it is correct and complete. The methodology has been implemented in projects Nova Desk, Nova Light, Nova Base and Nova Servers helping to eliminate deviations detected. Moreover, if properly implemented the methodology provides support to projects seeking to achieve CMMI level 2.

Keywords: Methodology, distribution of GNU / Linux, Nova.

ÍNDICE DE CONTENIDO

INTRODUCCIÓN……1
CAPÍTULO 1 FUNDAMENTOS TEÓRICOS DE LOS PROCESOS DE DESARROLLO DE DISTRIBUCIONES DE GNU/LINUX 9
Introducción 9
1.1 Análisis bibliométrico documental..... 9
1.2 Conceptualización de las distribuciones GNU/Linux 9
1.3 Proceso de desarrollo de distribuciones GNU/Linux 10
1.4 Métodos de desarrollo de software 13
1.4.1 Modelos de desarrollo de software 14
1.4.2 Metodologías de desarrollo de software 14
1.4.3 Comparación de las metodologías estudiadas 20
1.5 Calidad de Software..... 21
1.5.1 Consideraciones sobre la calidad de software..... 21
1.5.2 Programa de Mejora..... 21
1.6 Roles responsables del desarrollo de las distribuciones de GNU/Linux Nova 22
1.7 Infraestructura tecnológica del Departamento Sistemas Operativos para el desarrollo de distribuciones de GNU/Linux Nova 22
Conclusiones del Capítulo 1 23
CAPÍTULO 2 METODOLOGÍA PARA DESARROLLAR LA DISTRIBUCIÓN CUBANA GNU/LINUX NOVA..... 25
Introducción 25
2.1 ¿Qué es Nova - OpenUp? 25
2.1.1 Componentes de la Metodología Nova - OpenUp..... 25
2.1.2 Principios de la Metodología Nova - OpenUp 25
2.1.3 Prácticas de la Metodología Nova - OpenUp..... 26
2.1.4 Fases de Nova - OpenUp..... 26
2.1.5 Roles..... 27
2.1.6 Productos de trabajo 32
2.1.7 Disciplinas de Nova – OpenUp..... 33
2.1.8 Relación entre disciplinas, actividades, roles y artefactos..... 34
Conclusiones del Capítulo 2 47
CAPÍTULO 3 EVALUACIÓN DE LA METODOLOGÍA PARA EL DESARROLLO DE LA DISTRIBUCIÓN DE GNU/LINUX NOVA 48
Introducción 48
3.1 Evaluación de la variable independiente Metodología para desarrollar la distribución de GNU/Linux Nova..... 48
3.1.1 Selección de especialistas..... 48
3.1.2 Indicador Estado correcto de la metodología..... 50
3.1.3 Indicador Nivel de importancia de la metodología..... 51
3.1.4 Indicador Nivel de suficiencia de la metodología 52
3.2 Evaluación de la variable dependiente Desviaciones existentes en el proceso de desarrollo de la distribución GNU/Linux Nova 53
3.2.1 Indicador 1 Grado de utilización de una metodología de desarrollo de software…54
3.2.2 Indicador 2 Nivel de adecuación de los roles que se dedican al desarrollo de distribuciones GNU/Linux..... 54
3.2.3 Indicador 3 Nivel de adecuación de las actividades de gestión de requisitos….55
3.2.4 Indicador 4 Nivel de adecuación de las actividades de gestión de la configuración de

software. 55
3.2.5 Indicador 5 Nivel de adecuación de las actividades de dirección de proyectos.56
3.2.6 Indicador 6 Nivel de adecuación de las actividades de gestión de la calidad……. 57
3.2.7 Indicador 7 Grado de utilización del expediente de proyecto…… 57
3.3 Evaluación mediante el método SCAMPI C 59
3.4 Comparación de la metodología Nova – Open con otras propuestas similares…59
3.5 Análisis económico de la implementación de la metodología Nova – OpenUp….61
Conclusiones del Capítulo 3 63
CONCLUSIONES..... 64
RECOMENDACIONES 65
BIBLIOGRAFÍA 66
ANEXOS 71

INTRODUCCIÓN

El mercado de software ha dado en los últimos años un giro con la introducción de programas libres, brindando a los usuarios la libertad de poder distribuir, modificar, estudiar y mejorar las aplicaciones que utilizan. Muchas organizaciones han tomado la decisión de migrar a estándares de software libre[1] y código abierto[2], teniendo en cuenta que las aplicaciones de este tipo responden a costos menores de inversión, más seguridad y menos limitaciones en la automatización de sus procesos.

Uno de los productos más interesantes en este entorno son las distribuciones GNU/Linux, compuestas por una distribución del sistema operativo Linux, junto con un compilado de software, de tal forma que sea fácil de descargar, instalar y usar. Esto permite lograr un sistema perfectamente usable y con muchas funcionalidades [1].

Los departamentos pertenecientes al Centro CESOL[3] , de la Universidad de las Ciencias Informáticas (UCI) tienen el compromiso de apoyar la independencia tecnológica, respondiendo a que "…la migración de la informática en Cuba a sistemas operativos de código abierto (FOSS), debe proporcionar el aumento de la productividad y la agilidad para responder a cambios en el mercado, se reducirá la dependencia económico-comercial, influirá en la seguridad, se mantendrá una solución efectiva en costos y se ofrecerán mejores servicios o productos. Además, se reducirá la extraordinaria necesidad de adquirir las licencias privativas de software que ha obligado a operar la mayoría de las computadoras del país con programas que no se pagan y por los cuales, en caso de no existir el bloqueo económico impuesto por Estados Unidos a la isla, se deberían pagar cientos de millones de dólares…"[2]. Por tal razón, en el marco de la feria Informática 2009 es lanzada la versión 1.0 de la distribución de GNU/Linux Nova, la cual ha avanzado hoy hasta su versión 4.0.

El proceso de desarrollo de Nova debe estar enfocado a alcanzar: Niveles de excelencia en los aspectos de seguridad, mediante el modelo de desarrollo colaborativo, el acceso al código fuente, el exhaustivo proceso de revisión, la auditoría de código garantizando un sistema seguro de virus y sin puertas traseras. Soberanía Tecnológica proporcionando la formación de recursos humanos capacitados. Nova debe ser un sistema operativo independiente, con capacidad de decisión sobre las tecnologías reutilizadas y desarrolladas. Socio-adaptabilidad porque es un sistema alineado a las políticas que orienta la informatización nacional y optimizado para las condiciones tecnológicas del

1 Software Libre: Programa informático que cumple con 4 libertades expresadas en copiar, modificar, mejorar y redistribuir un software.

2 Código Abierto: Utiliza el software libre teniendo un punto de vista más orientado a los beneficios prácticos de poder acceder al código, que a las cuestiones éticas y morales.

3 CESOL: Centro de Software Libre.

país. Por último, sostenibilidad, pues debe mantener un proceso flexible y versátil, en constante innovación y consonancia con las nuevas tendencias tecnológicas internacionales, garantizando modelos de comercialización que permitan el ingreso de divisas por el concepto de exportación de productos y servicios [7]. A pesar de la relevancia que tiene este producto, en la segunda auditoría interna hecha por el proyecto Nova QALIT[4] se detectaron importantes desviaciones en el proceso de desarrollo de la distribución de GNU/Linux como son:

1) Cronogramas inexactos por varias causas, entre ellas requisitos que no son especificados correctamente y no tener en cuenta el ciclo de vida del proyecto, lo que provoca atrasos en las fechas de entrega y aumento de los costos estimados para el desarrollo del proyecto.
2) Está afectada la concepción de los requisitos; lo que se manifiesta en las dificultades y en el dominio de las relaciones entre las funcionalidades de software.
3) La gestión de la configuración de software se ejecuta de manera pobre, no se conoce ni tan siquiera este proceso por su nombre, sino que de manera inconsciente se lleva a cabo mediante el control de versiones.
4) Deficiente formación del personal nuevo que necesita conocer los procesos que ocurren y la manera de ejecutar determinadas actividades.
5) Incumplimiento de los estándares seleccionados, porque no cumplen los principios de la metodología SXP ni las políticas institucionales emitidas por CaliSoft[5].
6) Las actividades de gestión de la calidad existen a un nivel bajo, provocando que no se detecten y corrijan desviaciones a tiempo.
7) La documentación existente sobre los desarrollos es insuficiente porque no se utiliza adecuadamente el expediente de proyecto que propone CaliSoft. [3]

Después de realizar una entrevista (Ver anexo 1) a desarrolladores con experiencia de las variantes de la distribución GNU/Linux Nova, sobre las posibles causas de las desviaciones detectadas se pudo constatar la utilización incorrecta de la metodología SXP. Dicha metodología promueve un enfoque ágil exigente de un conjunto de condiciones como personal altamente capacitado, poca documentación, cliente en sitio, actividades mínimas de gestión de la calidad y de la configuración de software, etc. El desarrollo de la distribución GNU/Linux Nova no se ajusta a las condiciones antes mencionadas, porque los equipos de trabajo están medianamente capacitados, son inestables, no se trabaja con el cliente en sitio sino con una comunidad de usuarios dispersa y requiere de actividades

4 Nova QALIT: Proyecto que pertenece al departamento Sistemas Operativos que tiene como meta principal garantizar la calidad de los procesos y productos que tienen lugar en dicho departamento.
5 Dirección que se dedica a la Gestión de la Calidad.

que ayuden a la prevención y corrección de errores de manera constante. Sumando a lo antes expuesto, SXP obvia procesos importantes como el empaquetamiento, compilación y generación de repositorios. La situación problemática antes descrita, dada por la detección de las desviaciones identificadas por el proyecto Nova QALIT, posibilita definir el ***problema científico***: ¿Cómo contribuir a la disminución de las desviaciones en el proceso de desarrollo de la distribución cubana de GNU/Linux Nova? El ***objeto de estudio*** lo constituye el proceso de desarrollo de la distribución GNU/Linux Nova, determinando como ***objetivo de la investigación*** diseñar una metodología de desarrollo de software que contribuya a la disminución de las desviaciones en el proceso de desarrollo de la distribución cubana de GNU/Linux Nova.

Para cumplir con el objetivo de la investigación se definen los ***objetivos específicos***:

1) Determinar los sustentos teóricos que fundamentan el proceso de desarrollo de las distribuciones GNU/Linux.
2) Definir los componentes que debe tener una metodología para desarrollar distribuciones de GNU/Linux.
3) Validar la aplicación de la metodología de desarrollo de distribuciones GNU/Linux mediante la correlación de las variables descritas.

Estos inciden en el ***campo de acción*** las metodologías orientadas a los procesos de desarrollo de distribuciones de GNU/Linux. El ***tipo de investigación*** a desarrollar es explicativa, pretendiendo la identificación de desviaciones en el proceso de desarrollo de Nova, determinando las variables que inciden en su ocurrencia para solucionarlas.

La investigación se sustenta en la ***hipótesis***: Si se utiliza una metodología para desarrollar la distribución de GNU/Linux Nova, caracterizada por estar integrada al programa de mejora propuesto por CaliSoft en su versión 3.4, contener actividades básicas del trabajo en entornos de software libre e integrar procesos claves del desarrollo de distribuciones de GNU/Linux, esta contribuirá a la disminución de las desviaciones existentes en el proceso de desarrollo de la distribución GNU/Linux Nova en la UCI.

Se identifica la **variable independiente**: Metodología para desarrollar la distribución de GNU/Linux Nova, esta se define de manera conceptual y operacional (Tabla 1).

Definición conceptual: Guía de buenas prácticas que permite la correcta interacción entre los pilares fundamentales de un proceso de desarrollo de software. Incluye en su concepción elementos que favorecen la correcta obtención de las distribuciones de GNU/Linux Nova. Se sustenta en las características fundamentales: contener actividades básicas del trabajo en entornos de software libre

e integrar procesos claves del desarrollo de distribuciones de GNU/Linux.

Tabla 1: Indicadores de la variable independiente [Fuente: Elaboración Propia]

Indicadores	Definición conceptual	Escala de valores
Nivel de importancia de la metodología.	Conveniencia de utilizar la metodología para el desarrollo de la distribución GNU/Linux Nova.	***Alto***: Cuando entre el 70% y el 100% de los especialistas manifiesta que la metodología es importante. ***Medio***: Cuando entre el 50% y el 69% de los especialistas manifiesta que la metodología es importante. ***Bajo***: menos del 50% de los especialistas manifiesta que la metodología es importante.
Estado correcto de la metodología.	Estructura y composición adecuada de la metodología, teniendo en cuenta la existencia de todos los componentes de una metodología de desarrollo de software, y el cumplimiento de las características: integración al programa de mejora, trabajo en entornos de software libre e integración de procesos claves para el desarrollo de distribuciones de GNU/Linux.	***Alto***: Cuando entre el 70% y el 100% de los especialistas manifiesta que la metodología es correcta. ***Medio***: Cuando entre el 50% y el 69% de los especialistas manifiesta que la metodología es correcta. ***Bajo***: menos del 50% de los especialistas manifiesta que la metodología es correcta.
Nivel de suficiencia de la metodología.	Capacidad de la metodología de afrontar los pilares fundamentales del desarrollo de la distribución de GNU/Linux Nova.	***Alto***: Cuando entre el 70% y el 100% de los especialistas manifiesta que la metodología es suficiente. ***Medio***: Cuando entre el 50% y el 69% de los especialistas manifiesta que la metodología es suficiente. ***Bajo***: menos del 50% de los especialistas manifiesta que la metodología es suficiente.

La **variable dependiente** se expresa en: Desviaciones existentes en el proceso de desarrollo de la distribución de GNU/Linux Nova. A continuación se presentan sus definiciones conceptual y operacional (Tabla 2).

Definición conceptual: Condiciones que atentan contra la ejecución correcta del proceso de desarrollo de la distribución de GNU/Linux Nova.

Tabla 2: Indicadores de la variable dependiente [Fuente: Elaboración Propia]

Indicadores	Definición conceptual	Escala de valores
Grado de utilización de una metodología de desarrollo de	Medida en que los proyectos de desarrollo de distribuciones de GNU/Linux	***Alto***: Cuando entre el 70% y el 100% de los especialistas manifiesta utilizar totalmente la metodología de desarrollo de software.

software.	Nova utilizan la metodología de desarrollo de software.	***Medio***: Cuando entre el 50% y el 69% de los especialistas manifiesta utilizar parcialmente la metodología de desarrollo de software. ***Bajo***: Menos del 50% de los especialistas manifiesta utilizar la metodología de desarrollo de software.
Nivel de adecuación de los roles que se dedican al desarrollo de distribuciones GNU/Linux.	Medida en que los roles que propone la metodología cubren todas las responsabilidades a cumplir en el desarrollo de la distribución de GNU/Linux Nova.	***Alto***: Cuando entre el 70% y el 100% de los roles se encuentran representados en los procesos de desarrollo de Nova. ***Medio***: Cuando entre el 50% y el 69% de los roles se encuentran representados en los procesos de desarrollo de Nova. ***Bajo***: Menos del 50% de los roles se encuentran representados en los procesos de desarrollo de Nova.
Nivel de adecuación de las actividades de gestión de requisitos.	Cantidad de actividades básicas de gestión de requisitos que se realizan en los proyectos de desarrollo de distribuciones de GNU/Linux Nova, teniendo como referencia las propuestas por la metodología.	***Alta***: Cuando entre el 70% y el 100% de las actividades básicas propuestas para la gestión de requisitos se ejecutan en los proyectos de desarrollo de Nova. ***Media***: Cuando entre el 50% y el 69% de las actividades básicas propuestas para la gestión de requisitos se ejecutan en los proyectos de desarrollo de Nova. ***Baja***: Menos del 50% de las actividades básicas propuestas para la gestión de requisitos se ejecutan en los proyectos de desarrollo de Nova.
Nivel de adecuación de las actividades de gestión de la configuración de software.	Cantidad de actividades básicas de gestión de la configuración de software que se realizan en los proyectos de desarrollo de distribuciones de GNU/Linux Nova, teniendo como referencia las propuestas por la metodología.	***Alta***: Cuando entre el 70% y el 100% de las actividades básicas propuestas para la gestión la configuración de software se ejecutan en los proyectos de desarrollo de Nova. ***Media***: Cuando entre el 50% y el 69% de las actividades básicas propuestas para la gestión de la configuración de software se ejecutan en los proyectos de desarrollo de Nova. ***Baja***: Menos del 50% de las actividades básicas propuestas para la gestión de la configuración de software se ejecutan en los proyectos de desarrollo de Nova.
Nivel de adecuación de las actividades de dirección de proyectos.	Cantidad de actividades básicas de dirección de proyectos que se realizan en el desarrollo de distribuciones de GNU/Linux Nova, teniendo como referencia las propuestas por la metodología.	***Alta***: Cuando entre el 70% y el 100% de las actividades básicas propuestas para la dirección de proyectos se ejecutan en el desarrollo de Nova. ***Media***: Cuando entre el 50% y el 69% de las actividades básicas propuestas para la dirección de proyectos se ejecutan en el desarrollo de Nova.

		Baja: Menos del 50% de las actividades básicas propuestas para la dirección de proyectos se ejecutan en el desarrollo de Nova.
Nivel de adecuación de las actividades de gestión de la calidad de software.	Cantidad de actividades básicas de gestión de la calidad de software que se realizan en los proyectos de desarrollo de distribuciones de GNU/Linux Nova, teniendo como referencia las propuestas por la metodología.	***Alta***: Cuando entre el 70% y el 100% de las actividades básicas propuestas para la gestión de la calidad de software se ejecutan en los proyectos de desarrollo de Nova. ***Media***: Cuando entre el 50% y el 69% de las actividades básicas propuestas para la gestión de la calidad de software se ejecutan en los proyectos de desarrollo de Nova. ***Baja***: Menos del 50% de las actividades básicas propuestas para la gestión de la calidad de software se ejecutan en los proyectos de desarrollo de Nova.
Grado de utilización del expediente de proyecto.	Medida en que los desarrolladores dejan evidencia de sus procesos en el expediente de proyecto.	***Alto***: Se documenta en el expediente de proyecto. ***Medio***: Se documenta parcialmente en el expediente de proyecto. ***Bajo***: No se documenta en el expediente de proyecto.

Se utilizaron en el transcurso de la investigación **métodos científicos**. Dentro de los ***métodos teóricos*** se seleccionó la revisión bibliográfica, permite obtener información sobre los procesos de desarrollo de distribuciones de GNU/Linux y metodologías de desarrollo de software implementadas en varios proyectos. El método sistémico para estudiar las actividades que se llevan a cabo en estos desarrollos, observando las relaciones que existen entre ellas. De los ***métodos empíricos*** se utilizaron el Experimento y la Observación en su variante incluida. Estos permiten valorar la situación existente, determinando sus ineficiencias y la manera en que puede cambiar al introducir la solución propuesta (Ver anexo 2 Guía de observación). Además se utilizó el método Criterio de Expertos. Para recopilar información necesaria en la investigación se utilizaron las técnicas Entrevista (Ver anexo 1 Guía de observación) y Encuesta (Ver Anexos 2). La ***población*** está compuesta por el total de los ingenieros de software de la UCI. De ella se selecciona una ***muestra no probabilística*** representada por desarrolladores experimentados de la distribución de GNU/Linux Nova y especialistas en Ingeniería y Gestión de la calidad de software. La investigación responde a un diseño ***Experimental Puro*** que permite medir el efecto de la variable independiente sobre la dependiente.

El ***aporte práctico de la investigación*** se evidencia en una metodología para desarrollar la distribución cubana de GNU/Linux Nova evaluada bajo criterios de validación. Está incide directamente en el desarrollo y calidad de las variantes de la distribución cubana de GNU/Linux Nova. Para esto define un conjunto de procesos, productos de trabajo, roles e ilustraciones que la

convierten en material de estudio obligatorio para nuevos desarrolladores que ingresan al Departamento Sistemas Operativos.

La **novedad científica** de los resultados de esta investigación consiste en el diseño de una metodología para el desarrollo de distribuciones de GNU/Linux, basada en las características: alineación al programa de mejora versión 3.4 dictado por CaliSoft, trabajo en entornos de software libre e integración de procesos claves para el desarrollo de distribuciones de GNU/Linux, estableciendo como base para su concepción una metodología de desarrollo de software.

La tesis se estructura en tres capítulos. El ***Capítulo 1 Fundamentos teóricos de procesos de desarrollo de distribuciones de GNU/Linux***, describe pautas de las metodologías, modelos, estándares de calidad y procesos de desarrollo de software atendiendo al ciclo de vida de las distribuciones GNU/Linux. El ***Capítulo 2 Metodología para desarrollar la distribución cubana de GNU/Linux Nova***, presenta la estructura y componentes de la metodología Nova - OpenUp creada para desarrollar la distribución de GNU/Linux Nova. Por último, el ***Capítulo 3 Validación de la metodología para el desarrollo de la distribución de GNU/Linux Nova***, expone los resultados de evaluar si es correcta la metodología Nova - OpenUp de acuerdo al criterio de especialistas y de la implementación de porciones de la misma.

Relacionados con la actual investigación existe un ***listado de publicaciones, eventos y avales*** que se evidencian a continuación:

1. Fernández del Monte, Yusleydi. (2012) Metodología Nova – OpenUP para el desarrollo de distribuciones GNU/Linux. UCIENCIA 2012.
2. Fernández del Monte, Yusleydi. Guerrero Lambert, Sonia. Porro Lugo, Nadia. González Jorrin, Michael. Capacitación orientada a eliminar deficiencias en el aprendizaje. Sistemas de gestión de conocimientos. Universidad 2010.
3. Fernández del Monte, Yusleydi. Guerrero Lambert, Sonia. Modelo dML – UCI para gestionar conocimientos. 2011. Memorias del Evento XIV Convención y Feria Internacional, Informática.
4. Fernández del Monte, Yusleydi. Guerrero Lambert, Sonia. 2010. Modelo dML-UCI para gestionar conocimientos. Memorias de la Novena Semana Tecnológica de Fordes Las TIC: Presente y Futuro.
5. Fernández del Monte, Yusleydi. Guerrero Lambert, Sonia. 2010. Capacitación orientada a eliminar deficiencias en el aprendizaje. Sistema de gestión de conocimientos. Memorias del Taller Internacional de Pedagogía de la Educación Superior.
6. Fernández del Monte, Yusleydi. 2012 ¿Cómo desarrollar distribuciones GNU/Linux cumpliendo con las buenas prácticas de la ingeniería de software? Memorias del Evento UCIENCIA 2012.

7. Pérez Herrera, Dariem. Guerrero Lambert, Sonia. Fernández del Monte, Yusleydi. Albalat Águila, Miguel. Machín, Jorge Luis. Pérez Baranda, Héctor. Quevedo Mejías, Ricardo. 2012. Estado actual de los sistemas de construcción de paquetes en diferentes distribuciones de GNU/Linux. Revista Cubana de Ciencias Informáticas.
8. Fírvida Donéstevez, Abel Alfonso. Albalat Aguila, Miguel. Hurtado Fedorovich, Mijail. Soler Franco, Yunier. Machin Castillo, Jorge Luis. Miranda Gómez, Raydel. Pierra Fuentes, Allan. Pérez Herrera, Dariem. Goñi Orama, Angel. Herrera, Anielkis. Fernández Del Monte, Yusleydi. 2012. Nova 3.0, avances y expectativas de la distribución cubana de GNU/Linux. Serie Científica de la Universidad de las Ciencias Informáticas.
9. Fernández del Monte, Yusleydi. (2012) Curso de Ingeniería de Software Libre. XVI Fórum de Ciencia y Técnica, Evento Municipal.
10. Fírvida Donéstevez, Abel Alfonso. Albalat Aguila, Miguel. Hurtado Fedorovich, Mijail. Soler Franco, Yunier. Machin Castillo, Jorge Luis. Miranda Gómez, Raydel. Pierra Fuentes, Allan. Pérez Herrera, Dariem. Goñi Orama, Angel. Herrera, Anielkis. Fernández Del Monte, Yusleydi. 2012. Nova 3.0, avances y expectativas de la distribución cubana de GNU/Linux. 2012. Memorias del Taller Temático de Fordes Formación para la migración a estándares de código abierto
11. Fírvida Donéstevez, Abel Alfonso. Albalat Aguila, Miguel. Hurtado Fedorovich, Mijail. Soler Franco, Yunier. Machin Castillo, Jorge Luis. Miranda Gómez, Raydel. Pierra Fuentes, Allan. Pérez Herrera, Dariem. Goñi Orama, Angel. Herrera, Anielkis. Fernández Del Monte, Yusleydi. 2011. Nova 3.0, avances y expectativas de la distribución cubana de GNU/Linux. Memorias del Evento XIV Convención y Feria Internacional, Informática 2011
12. Fernández del Monte, Yusleydi. 2010. Modelo dML-UCI para gestionar conocimientos. 2012. ISSN: 2076-9792. Memorias de la Novena Semana Tecnológica de FORDES.
13. Fernández del Monte, Yusleydi. (2008) Capacitación orientada a eliminar deficiencias en el aprendizaje. Sistemas de gestión de conocimientos. ISBN: 978-959-286-007-0 Memorias del Evento UCIENCIA 2008.
14. Fernández del Monte, Yusleydi. Propuesta de la metodología Nova OpenUp para el desarrollo de distribuciones GNU/Linux. 2012. Memorias del evento III Taller Nacional de Software Libre Presente y Futuro.
15. Fernández del Monte, Yusleydi. Ingeniería de Software Libre. 2012. Memorias del evento III Taller Nacional de Software Libre Presente y Futuro.

CAPÍTULO 1 FUNDAMENTOS TEÓRICOS DE LOS PROCESOS DE DESARROLLO DE DISTRIBUCIONES DE GNU/LINUX

Introducción

En este capítulo se le da respuesta a las siguientes interrogantes: ¿Qué es una distribución GNU/Linux? ¿Qué procesos de desarrollo siguen las distribuciones GNU/Linux? ¿Cómo obtener calidad en el proceso de desarrollo de software? ¿Qué roles son necesarios para construir distribuciones GNU/Linux y qué productos de trabajo ellos pueden generar? ¿Qué metodología de desarrollo de software sería adecuada para desarrollar la distribución de GNU/Linux Nova?

1.1 Análisis bibliométrico documental

En el análisis bibliométrico (ver Tabla 3) se refleja la revisión bibliográfica de la literatura científica para identificar las principales fuentes y áreas de conocimiento que engloba el desarrollo de distribuciones de GNU/Linux. Es significativo que los proyectos que se dedican a obtener este tipo de producto no muestren una metodología o modelo que integre todos los procesos que ejecutan. Se analizan teorías relacionadas con las metodologías y modelos de desarrollo de software que sirven a la autora para innovar en su adaptación al entorno de desarrollo de la distribución GNU/Linux Nova. Además, se muestra que la bibliografía consultada es amplia y actualizada, con abundante uso de recursos consultados en artículos publicados en la Web; así como tesis de maestrías, reportes técnicos de proyectos investigativos-productivos y libros.

Tabla 3: Análisis bibliométrico documental [Fuente: Elaboración Propia]

	Últimos 5 años (%)	**Años anteriores (%)**
Libros y monografías	9	18
Tesis de maestrías	4	0
Tesis de doctorado	0	0
Arículos en Revistas referenciadas en Web of Science, SCOPUS	4	0
Memorias de eventos	2	0
Artículos publicados en la web	36	11
Reportes técnicos y conferencias	13	2
Entrevistas personales	4	0

1.2 Conceptualización de las distribuciones GNU/Linux

Las distribuciones GNU/Linux son creadas para satisfacer necesidades concretas y con un objetivo específico, teniendo en cuenta factores de calidad como la seguridad, eficiencia, usabilidad y funcionalidad.

A partir de las definiciones consultadas en las bibliografías [4], [5], [6], [7], [8] se puede decir que las

distribuciones GNU/Linux responden a una arquitectura basada en componentes. Están caracterizadas por un conjunto de funcionalidades y restricciones que responden a necesidades específicas de un grupo de personas. Pueden ser orientadas a usuarios con conocimientos elementales, a empresas o a servidores. Están compuestas por un sistema instalable que contiene todas las aplicaciones y configuraciones de acuerdo a las metas establecidas y un repositorio que permite instalar y desinstalar paquetes de código binario.

1.3 Proceso de desarrollo de distribuciones GNU/Linux

Realizando un estudio de la gran cantidad de proyectos de desarrollo de distribuciones GNU/Linux (más 60) se determina estudiar el proceso de desarrollo de Debian por su antigüedad y muestra de elevado nivel de madurez y estabilidad. Se tuvo en cuenta que de Debian descienden las distribuciones Ubuntu y de esta Nova. Se selecciona Ubuntu y Fedora porque según reportes del sitio Linux.com (muestra las 7 mejores distribuciones por años) en el 2012 estos fueron los mejores sistemas de escritorio. Por último Nova porque tiene gran relevancia en esta investigación. Los proyectos seleccionados coinciden con ser bastante divulgados y utilizados en el entorno nacional. Profundizando en las características de cada proceso de desarrollo se obtienen las siguientes descripciones:

Guía Linux from Stratch (LFS) versión 7.0 hace referencia a cómo instalar un sistema operativo GNU/Linux desde cero, brindando a usuarios con conocimientos elementales las opciones de conocer su funcionamiento, controlar el sistema pudiendo dictar el aspecto del mismo, crear un sistema compacto instalando solo lo necesario, además de brindar más seguridad porque personalmente se puede compilar todo, supervisarlo y aplicar los debidos parches al código fuente [9].

Proceso de desarrollo de la Distribución GNU/Linux Debian, dicha distribución pertenece a uno de los proyectos más reconocidos en esta área, posee un gran equipo de desarrollo distribuido en varias comunidades del mundo, todos tienen por meta fundamental, obtener un sistema operativo libre garantizando la calidad, seguridad y continuidad del producto. Para esto se compone de varios proyectos que garantizan la distribución, comunicación, documentación, control de la calidad, seguridad, almacenamiento, divulgación e intercambio, permitiendo así obtener la distribución GNU/Linux Debian. Herramientas CASE (Ingeniería de Software Asistida por Computación) que soportan este proceso son Patch, Diff, Configure, Make, Scripts en Shell, y otras de empaquetado como dpkg-dev, debhelper, debconf, litian, linda y pbuilder. Los requisitos que debe tener la versión a lanzar se recopilan muchas veces de un sistema de seguimiento de fallas que da información sobre la depuración, pruebas, parcheo, peticiones de la infraestructura, mejoras y gestión de paquetes [10]. Los roles que muestra Debian en su constitución del proyecto versión 1.2 son desarrolladores

(realizan fundamentalmente tareas de mantenimiento de paquetes de código), líder de proyecto, secretario del proyecto, presidente del comité técnico y delegados del líder de proyecto. La gestión de la configuración de software se hace mediante la herramienta Subversión que permite controlar las versiones de los diferentes paquetes de código que los contribuidores suben al repositorio.

***Proceso de desarrollo de la Distribución GNU/Linux Ubuntu**,* popular distribución que tiene sus potencialidades en hacer un producto acorde a las experiencias de los usuarios, convirtiendo sus necesidades en requisitos a cumplir mejorando así la calidad de uso. Uno de los procesos que potencia el desarrollo de este producto son los enfocados a la calidad de software garantizando que los paquetes respondan a los estándares que ellos definan. La toma de decisiones en los procesos de gestión de proyectos es llevada a cabo por el Community Council de Ubuntu, equipo de gobierno principal encargado de tomar las principales decisiones. Dan principal importancia a los procesos de documentación de las aplicaciones y a la disponibilidad de la información necesaria para hacer a los contribuidores parte de sus desarrollos. En sus planificaciones contemplan varias iteraciones para la liberación de las versiones alpha, beta y release candidate de sus productos. Fomentan el intercambio con su comunidad de usuarios a través de sistemas de seguimiento de errores [11]. La gestión de la configuración de software se hace mediante la herramienta Subversión que permite controlar las versiones de los diferentes paquetes de código que los contribuidores suben al repositorio.

***Proceso de desarrollo de la Distribución GNU/Linux Fedora**,* pertenece a la empresa Red Hat, se caracteriza por tener un ciclo de desarrollo rápido, la gestión de proyectos es liderada por 9 miembros de ellos 4 son elegidos por la comunidad y el resto por los patrocinadores de Red Hat. Cada subproyecto tiene su propio comité de dirección responsable de tomar las decisiones que tienen que ver con sus tareas. Estos subproyectos se dedican al desarrollo de herramientas gráficas, detección, seguimiento de errores, documentación de procesos y herramientas, creación de juegos, definición de estándares de usabilidad y empaquetamiento de software. Roles definidos para el desarrollo de este producto son: - Escritor de contenido, - Diseñador gráfico, - Contacto con personas, - Desarrollador del sistema, - Traductor y - Desarrollador web o Administrador [12]. La gestión de la configuración de software se hace mediante la combinación de las herramientas Subversión y Git que permiten controlar las versiones de los diferentes paquetes de código.

***Proceso de desarrollo de la Distribución GNU/Linux Nova**,* este sistema utiliza el núcleo de Linux e incluye determinados paquetes de aplicaciones informáticas para satisfacer las necesidades de la migración a plataformas de código abierto que experimenta Cuba como parte del proceso de informatización de la sociedad [13]. El proceso de desarrollo para la distribución GNU/Linux Nova en

el período 2010 – 2012 ocurría teniendo en cuenta las 4 fases que plantea la metodología SXP:

Planificación - Definición cuya meta era establecer la visión, fijar las expectativas y obtener financiamiento para el proyecto. Desarrollo con el propósito de obtener un sistema listo para entregar. Para esto las principales actividades estaban encaminadas a definir cada iteración, estas compuestas de tareas de planificación, diseño de software, codificación de funcionalidades y ejecución de pruebas de aceptación. Entrega basada en la puesta en operación del producto, incluyendo formas de documentación del sistema, formación de los usuarios involucrados e instalación en el ambiente real. Finalmente Mantenimiento, compuesta por la ejecución de actividades de soporte para el cliente. [14]

El proceso de desarrollo de la distribución GNU/Linux Nova en función de la metodología SXP ha tenido varias insuficiencias porque no está definida la correspondencia entre las actividades reales de la metodología SXP y las que realmente ocurren al desarrollar la distribución de GNU/Linux Nova. Ocurre que las actividades se hacen de manera artesanal guiándose por pautas genéricas que propone la metodología SXP sin adaptarlas a la realidad, lo que provoca continuas desviaciones en el cumplimiento de los objetivos del proyecto. La mayoría del personal involucrado en el desarrollo de este producto es novato e inestable, acorde con el plan de formación profesional de los estudiantes de la UCI cada año se va de estos proyectos personas con cierto nivel de capacitación e ingresan estudiantes totalmente inexpertos. Sumando, que los artefactos que propone SXP no se ajustan a los definidos por el programa de mejora dictado por CaliSoft. Los roles propuestos por esta metodología no tienen asignadas responsabilidades relacionadas con la compilación, empaquetamiento y generación de repositorios de paquetes de código. Como toda metodología ágil supone trabajar con un equipo altamente calificado y autónomo, por lo que las actividades de aseguramiento de la calidad, dirección de proyectos y gestión de la configuración (solo controlar versiones de productos software) propuestas son insuficientes teniendo en cuenta las características del desarrollo de la distribución GNU/Linux Nova.

En la investigación se observa internamente el proceso de desarrollo de la distribución GNU/Linux Nova, los demás procesos de desarrollo analizados solo muestran la comunicación con agentes externos como son las comunidades de usuarios. No está pública la metodología utilizada por los proyectos Debian, Fedora y Ubuntu; estos muestran en su sitio de alojamiento una serie de procesos necesarios para crear la distribución de GNU/Linux pero no se observa integración entre ellos. Es común en proyectos que desarrollan productos de este tipo la utilización de herramientas colaborativas para establecer comunicación entre la comunidad, clientes y desarrolladores. La dirección de proyectos que se lleva a cabo es tanto centralizada como descentralizada, dando control a cada quien sobre las tareas que le corresponden, pero a la vez aceptando al desarrollo oficial lo que

apoya el cumplimiento de los objetivos establecidos. Una de las fuentes principales para la obtención de requisitos sugeridos por los usuarios son los registros en herramientas de gestión de incidencias como Bugzilla. Entorno al desarrollo de distribuciones GNU/Linux se crean otros proyectos que les sirven de apoyo, ejemplo son los que se dedican a la documentación de las aplicaciones, divulgación de las tareas, a las estrategias para hacer crecer la comunidad de usuarios, garantizar la retroalimentación necesaria, la gestión de la configuración, la calidad y el seguimiento de errores. Algunos de estos proyectos potencian la documentación de procesos y aplicaciones garantizando su continuidad en el tiempo. Uno de los roles que más se destaca es el llamado desarrollador que tiene entre sus competencias principales el empaquetamiento y compilación de paquetes de código. La gestión de la configuración de software se lleva a cabo de manera tanto centralizada como distribuida utilizando como apoyo herramientas como Subversión y Git.

En los procesos de desarrollo de distribuciones de GNU/Linux analizados, no se evidencia la utilización de una guía que integre todos los procesos necesarios para obtener este producto desde su concepción hasta su terminación. En el caso particular de las LFS, solo muestra actividades orientadas al desarrollo del producto, obviando las que deben ser orientadas al proceso.

1.4 Métodos de desarrollo de software

Los procesos de desarrollo de software tienen 4 pilares fundamentales: personas, procesos, productos y proyecto, siendo este último la unidad organizativa que agrupa a todos, por tanto, un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único [15]. De manera general el ciclo de vida de un proyecto se representa como se puede ver en la Figura 1.

¿Qué hacer para desarrollar software? Es una interrogante cotidiana de empresas, “la experiencia acumulada a lo largo de años de ejecutar proyectos, indica que los proyectos exitosos son aquéllos que son administrados siguiendo una serie de procesos que permiten organizar y luego controlar al proyecto” [16].

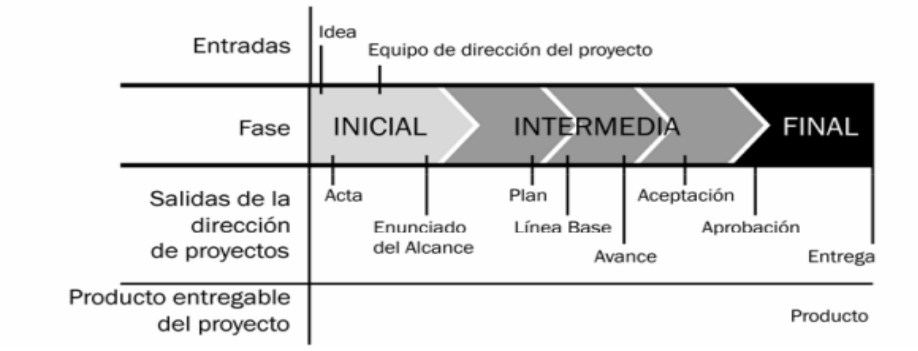


Figura 1 Ciclo de vida de un proyecto según PMBOK [15]

1.4.1 Modelos de desarrollo de software

Se estudiaron los modelos:

Cascada que sugiere un enfoque sistemático y secuencial hacia el desarrollo de software. Está demostrado que su naturaleza lineal conduce a estados de bloqueo pues se debe esperar repetidas veces que acabe una actividad para comenzar otra.

Incremental representa una variación del modelo en Cascada, pero sugiere la división del problema principal en pequeños subproblemas y la solución a cada uno resulta en un incremento de la solución para el problema principal.

V expresa una variación del modelo en cascada, vinculando las fases de desarrollo con acciones de aseguramiento de la calidad.

Evolutivo es iterativo y cada iteración resulta en una versión completa del producto, son variaciones de este, los modelos ***Prototipos*** y ***Espiral***.

Por último, ***Concurrente*** es aplicado a cualquier tipo de desarrollo de software y plantea que cada actividad, acción, o tareas están conectadas simultáneamente con otras actividades, acciones, o tareas. [17]

El análisis de estos modelos determina que el desarrollo de la distribución de GNU/Linux Nova responde al modelo Espiral (Ver Anexo 6). De manera iterativa se recorren varias espirales donde cada una de ella representa un incremento o mejora del sistema.

1.4.2 Metodologías de desarrollo de software

Según Avison y Fitzgerald, una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología esta formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo [17].

Por las diferencias que hay entre los ciclos de vida de los diferentes proyectos han surgido variedad de metodologías de desarrollo de software, cada una debe ser suficiente para lograr los objetivos establecidos. Para elegir la metodología adecuada se deben tener en cuenta un conjunto de factores que indican si es más apropiado llevar un proceso ligero o uno pesado, los mismos se ilustran en la tabla 4.

Tabla 4: ¿Qué enfoque de desarrollo utilizar? [17]

Factor	**Territorio Moderno**	**Territorio Tradicional**
Tamaño	Producto y proyecto pequeño.	Grupos y productos grandes, difíciles de

		ajustar a algo pequeño.
Criticidad	No están involucradas pérdidas humanas, por la sencillez se utilizan pocos métodos de aseguramiento de la calidad y documentación.	Productos críticos, métodos evolucionados para soportar la criticidad.
Dinamismo	Entornos donde los requisitos y la tecnología cambian de forma seguida.	Entornos estables.
Personal	La mayoría de las personas tienen altos conocimientos y experiencia en el tipo de proyecto a desarrollar.	La mayoría de las personas tienen pocos conocimientos y experiencia en el tipo de proyecto a desarrollar.
Cultura	Trabajo en equipo con bajo nivel control, dando bastantes libertades a los desarrolladores.	Trabajo en equipo bajos políticas, roles, tareas y procesos bien definidos.

Para elegir desarrollar con una metodología se deben tener en cuenta un conjunto criterios que apoyan el entendimiento y éxito de aplicarla, pues la misma debe cumplir con mayor presencia en internet, mejor documentación, certificación y entrenamiento, tener comunidades y ser suficientemente utilizada por empresas.

Teniendo en cuenta lo antes mencionado, más las características del proceso de desarrollo de las variantes de la distribución GNU/Linux Nova (puntos negros en la Figura 2) se debe tender más a un enfoque híbrido, combinando características tanto de las metodologías tradicionales como de las modernas. El proceso de desarrollo de Nova debe contener características de enfoques pesados, teniendo en cuenta que representa a un proyecto largo que tiene un tiempo mínimo de duración de 1 año, su alcance está bien definido, tiene en cuenta el cumplimiento de restricciones institucionales y posee recursos medianamente disponibles. Por otro lado, responde al desarrollo de un producto grande y complejo. Los equipos de desarrollo de Nova por lo general son inexpertos, con mediana experiencia en el dominio de la aplicación y medianamente autónomos. Debe incluir también características del enfoque ligero, porque se trabaja con equipos pequeños compuestos por una cantidad de 4 a 8 personas, el dominio de aplicación responde a software de línea de productos favoreciendo esto el continuo intercambio con la comunidad de usuarios. El equipo de desarrollo está altamente comprometido, tiene experiencia en el trabajo en equipo, además el grado de distribución y comunicación es alto. Por tal razón, en el estudio se tienen en cuenta metodologías tanto ligeras como pesadas. De las metodologías estudiadas ninguna se ajusta al desarrollo de Nova pero es válido conocer sus características generales para determinar cuál aporta más al objetivo de la investigación y cuál puede brindar buenas prácticas a tener en cuenta.

Las metodologías de desarrollo de software se pueden basar en un modelo o combinación de estos,

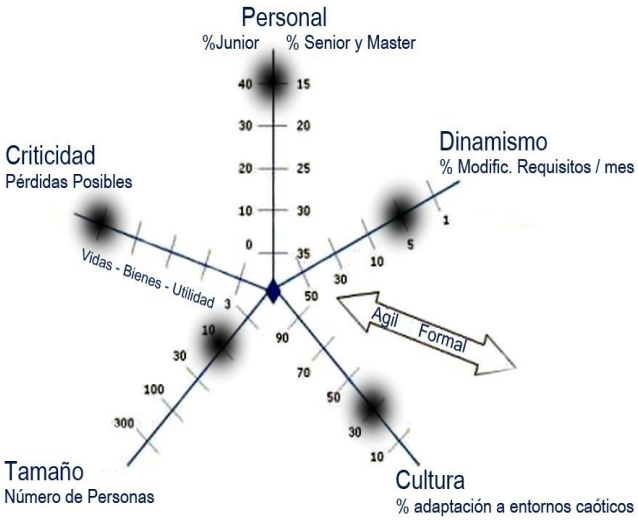


Figura 2 Gráfico para la selección de los modelos de desarrollo propuesto por Barry Boehm y Richard Turner

ellos indican una secuencia de pasos a seguir. Estos pasos guían a la metodología en la forma de aplicar un conjunto de métodos para cumplir con las metas que se persiguen. A continuación se hace un estudio de un conjunto de metodologías que responden al enfoque del modelo seleccionado en el epígrafe anterior.

Rational Unified Process (RUP), tiene por autores principales a Ivar Jacoboson, Grady Booch y James Rumbaugh. Es un proceso de ingeniería del software tradicional que proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades de una organización de desarrollo. Su propósito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales, con unos costos y calendarios predecibles [18]. Tiene tres elementos fundamentales, un conjunto subyacente de filosofías y principios para conseguir un desarrollo de software correcto, una infraestructura de bloques de construcción del proceso y contenido del método reutilizables, el método subyacente y el lenguaje de definición del proceso [19]. RUP es guiada por casos de uso, centrada en la arquitectura y plantea que el proceso de desarrollo debe dividirse en ciclos que sucedan de manera iterativa e incremental.

eXtreme Programming (XP), fue creada por Kent Beck. Es una metodología moderna que "actualmente es una de las más exitosas; esto se debe a que está centrada en potenciar las relaciones entre personas como clave para obtener éxito en el desarrollo de software. Promueve el trabajo en equipo, la comunicación con el cliente, proporcionando a cada uno de estos un buen clima de trabajo, y un aspecto muy importante es que se preocupa por el aprendizaje de los programadores" [20]. Las características fundamentales de XP están dadas por: desarrollo iterativo e

incremental, pruebas unitarias continuas, integración del equipo de programación con el usuario, corrección de todos los errores, refactorización del código, propiedad del código compartida.

Scrum, fue presentada formalmente para el desarrollo de software en 1996 por Ken Schwaber y Jeff Sutherland. “Es una metodología de desarrollo muy simple, que requiere trabajo duro, porque la gestión no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto [21]” Sus características fundamentales son: modo de desarrollo de carácter adaptable, orientado a las personas antes que a los procesos, y emplea desarrollo ágil, iterativo e incremental. Es útil en entornos con incertidumbre e inestabilidad de requisitos. Cada fase de trabajo con Scrum pasa por las actividades planificación de Sprint, seguimiento del Sprint, y revisión del Sprint.

Feature Driven Development (FDD), en idioma español Desarrollo Basado en Funcionalidades. Fue creada por Jeff De Luca y Peter Coad a mediados de los años 90. Es un enfoque ágil para el desarrollo de sistemas. Dicho enfoque no hace énfasis en la obtención de los requerimientos sino en cómo se realizan las fases de diseño y construcción. Sin embargo, fue diseñado para trabajar con otras actividades de desarrollo de software y no requiere la utilización de ningún modelo de proceso específico. Además, hace énfasis en aspectos de calidad durante todo el proceso e incluye un monitoreo permanente del avance del proyecto. Al contrario de otras metodologías, FDD afirma ser conveniente para el desarrollo de sistemas críticos [22]. Responde a un modelo Incremental caracterizado por la ejecución de iteraciones cortas que duren de una a dos semanas.

Crystal “Es un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo. Han sido desarrolladas por Alistair Cockburn de la IBM. Cockburn manifiesta que diferentes tipos de proyectos requieren diferentes tipos de metodologías razón por la cual crea la familia Crystal”. Los métodos Crystal no prescriben las prácticas de desarrollo, las herramientas o los artefactos que pueden usarse, pudiendo combinarse con otros métodos como Scrum, XP y Microsoft Solutions Framework. Esta metodología se caracteriza por entregas frecuentes mediante la ejecución de un modelo Incremental, comunicación íntima, mejora reflexiva, seguridad personal, enfoque, acceso fácil a los usuarios especialistas, ambiente técnico con pruebas automatizadas, administración de la configuración e integración frecuente [23].

Microsoft Solution Framework (MSF), es propiedad de Microsoft por lo que se plantea que su implementación es cara. Describe las mejores prácticas en términos de principios básicos, modelos conceptuales y disciplinas. Incluye dos metodologías prescriptivas: MSF para el desarrollo de aplicaciones ágiles y MSF para el proceso de mejora CMMI. Tiene como principios fundamentales:

promover comunicaciones abiertas, trabajar para una visión compartida, fortalecer los miembros del equipo, establecer responsabilidades claras y compartidas, focalizarse en agregar valor al negocio, permanecer ágil, y esperar los cambios, invertir en calidad, aprender de todas las experiencias, trabajar en parejas con clientes y siempre crear productos entregables. Responde a los modelos de desarrollo de software Cascada y Espiral.

OpenUp (Open Unified Process en español Proceso Unificado Abierto), fue donada en el año 2007 a la fundación Eclipse por un conjunto de empresas de tecnología. Es el resultado de adaptar el proceso unificado enfocándose en aspectos que la hacen más ágil y recomendada para ser utilizada en proyectos pequeños. Esta mantiene las características fundamentales de RUP haciendo una selección de las disciplinas y artefactos obligatorios. Divide los roles en tres grupos: roles básicos (Analista, Cualquier rol, Arquitecto, Desarrollador, Administrador de proyecto, Stakeholder y Probador), roles de ambiente (Ingeniero de proceso y Especialista de herramientas) y roles de despliegue (Desarrollador de cursos, Ingeniero de despliegue, Administrador de despliegue, Propietario del proyecto, Escritor técnico y Entrenador).

Los principios que propone están dados por equilibrar las prioridades para maximizar el valor de las partes interesadas, colaborar para sincronizar intereses, centrarse en la construcción de principios para minimizar los riesgos, organizar el desarrollo y evolucionar continuamente obteniendo retroalimentación y mejora.

La ejecución de las actividades que propone gira entorno a áreas de conocimientos agrupadas en prácticas de administración, técnicas y de despliegue.

Las prácticas de administración son: desarrollo iterativo, ciclo de vida basado en valor y riesgo, planificación del desarrollo del producto, equipo completo en función de las actividades a realizar, administración de cambios y adaptación de los procesos al entorno del proyecto.

Las prácticas técnicas son: pruebas concurrentes, integración continua, arquitectura y diseño evolutivo, visión compartida, pruebas y los casos de uso guían el desarrollo de software.

Las prácticas de despliegue son: documentación, entrenamiento y producción de release.

Las siguientes tablas muestran las actividades y artefactos agrupados por las disciplinas de esta metodología.

Tabla 5: Disciplinas ingenieriles [Fuente: Elaboración Propia]

Disciplinas	**Actividades**	**Artefactos**
Requisitos	Identificar y refinar requisitos Detallar escenarios de los casos de uso Detallar requisitos no funcionales Desarrollar la visión técnica	Glosario de términos Visión del Proyecto Requisitos no funcionales Modelo de Casos de Uso
Arquitectura	Definir arquitectura	Cuaderno de arquitectura

	Refinar arquitectura	
Desarrollo	Diseñar la solución Integrar y crear build Ejecutar pruebas del desarrollador Implementar solución	Build Implementación Diseño Pruebas de desarrollador
Prueba	Crear casos de prueba Implementar pruebas Registrar pruebas	Caso de prueba Script de prueba Registro de la prueba

Tabla 6: Disciplinas de apoyo [Fuente: Elaboración Propia]

Disciplinas	**Actividades**	**Artefactos**
Gestión de proyectos	Planificar proyecto Planificar iteración Administrar iteración Asegurar los resultados Responder a los cambios	Lista de Riesgos Lista de elementos de trabajo
Ambiente	Implementar el proceso Adaptar el proceso de configuración de herramientas Verificar la configuración e instalación de herramientas	Definición de los procesos del Proyecto Herramientas
Despliegue	Desarrollar la documentación del producto Desarrollar la documentación del usuario Desarrollar la documentación de soporte Formar usuarios Entregar formación de soporte Desarrollar materiales de capacitación Planificar lanzamiento Ejecutar el plan de restitución (si es necesario) Ejecutar el Plan de Despliegue Instalar y validar Infraestructura de implementación del plan	Documentación del producto Documentación de soporte Documentación de usuario Materiales de entrenamiento Plan de deshecho Plan de despliegue Infraestructura Release de comunicación Release de controles *Release*

SXP responde al desarrollo iterativo e incremental, fue creada por Gladys M. Peñalver Romero. Ofrece una estrategia tecnológica, basada en los valores y principios de las metodologías ágiles expuestos en el Manifiesto Ágil. Consta de 4 fases principales y de cada una de ellas se despliegan 7 flujos de trabajo: concepción inicial, captura de requisitos, diseño con metáforas, implementación, prueba, entrega de la documentación, soporte e investigación, el cual se utiliza por el equipo de desarrollo cuando sea necesario [24].

Ninguna de las metodologías estudiadas se alinea a las exigencias de la organización a donde pertenece el desarrollo de Nova, no cumplen con las políticas del programa de mejora, exceptuando a MSF que tiene una variante que integra al modelo CMMI, pero sin embargo es privativa. Además obvian procesos claves para el desarrollo de distribuciones de GNU/Linux, no definen todos los roles

necesarios y muchas no incluyen prácticas para el desarrollo en entornos de software libre.

1.4.3 Comparación de las metodologías estudiadas

Después de estudiar los detalles de las metodologías seleccionadas se hace una comparación entre ellas con la finalidad de elegir cuál sería la correcta para desarrollar la distribución GNU/Linux Nova. A cada criterio de evaluación se le otorga un valor entre 0 y 5, donde 5 significa mayor aceptación y 0 ninguna aceptación. Para otorgar cada nota se tiene como principal base las características del equipo de desarrollo de la distribución de GNU/Linux Nova del departamento Sistemas Operativos y la información disponible en internet sobre las metodologías estudiadas.

La metodología más adecuada según los resultados de la tabla 7 es OpenUp. OpenUp define un proceso de desarrollo mínimamente suficiente, los que trabajan con metodologías ágiles la consideran pesada y los que trabajan con metodologías modernas la consideran tradicional. Es completa, configurable y extensible para utilizarse como base en cualquier entorno. No se debe utilizar tal como es, porque no contiene elementos claves del desarrollo de las distribuciones de GNU/Linux Nova. La nueva variante de la metodología OpenUp debe sufrir transformaciones porque se sigue un enfoque más guiado al desarrollo pesado pero pueden existir determinados productos como personalizaciones de las distribuciones GNU/Linux Nova que exigen un desarrollo totalmente ágil. Se deben modificar sus actividades, roles y artefactos de acuerdo al proceso de desarrollo de la distribución GNU/Linux Nova, porque OpenUp no trata temas como el empaquetamiento, compilación de paquetes, generación de repositorios, etc. - La nueva variante debe pasar de ser guiada por caso de uso a ser guiada por requisitos, técnica más apropiada para describir las funcionalidades de una distribución GNU/Linux.

Tabla 7: Comparación de Metodologías de Desarrollo de Software [Fuente: Elaboración Propia]

Metodologías (Total de puntos)	Presencia en Internet	Más documentada	Certificada y con entrenamiento	Con comunidades	Más Utilizada	Soporte para personal inexperto	Criticidad	Cultura	Cambios continuos en el cronograma	Equipos de proyecto pequeños	Soporte para Distribución de GNU/Linux
RUP (48)	5	5	5	5	5	5	5	5	4	2	2
XP (39)	4	5	3	4	4	2	2	3	4	5	3
Scrum (36)	5	2	5	5	4	2	1	2	3	5	2
FDD (33)	5	3	1	2	3	2	2	3	4	5	3
Crystal (42)	5	4	2	3	3	4	4	4	5	5	3

MSF (33)	5	3	2	2	2	2	2	3	4	5	3
OpenUp (52)	5	5	5	5	4	5	5	5	5	5	3
SXP (32)	2	5	2	3	1	2	2	3	4	5	3

1.5 Calidad de Software

1.5.1 Consideraciones sobre la calidad de software

En la actualidad los procesos de gestión de la calidad de software se han convertido en una necesidad para las empresas, no se conciben en el mercado ofertas que no cumplan con los requisitos exigidos por los clientes. Por tal motivo, se hace necesario llevar a cabo actividades de garantía de la calidad desde que se concibe el software hasta su muerte como parte de las decisiones estratégicas de la organización.

La calidad en los procesos de desarrollo de software debe ser vista como lo plantea la guía de conocimientos para ingeniería de software (SWEBok), la cual expresa: "... Las características de calidad son definidas por los requisitos de software y estos influencian los métodos de medidas y criterios de aceptación para asegurar esas características..." La calidad debe verse como una cultura de trabajo por parte de los ingenieros de software. Para mejorar la calidad deben tenerse en cuenta los procesos del ciclo de vida del software, de detección, eliminación y prevención de errores o defectos, así como los procesos de mejora de la calidad. Todo lo antes dicho debe hacerse teniendo en cuenta al cliente. Esta área propone tareas y técnicas que permiten ver cómo se cumplen los planes y requisitos del software teniendo en cuenta la gestión de riesgos [25].

1.5.2 Programa de Mejora

El proceso de mejora está encaminado a que la UCI alcance una certificación internacional del nivel 2 del modelo CMMI. CMMI es un modelo de referencia para el crecimiento de capacidades y madurez, que se enfoca tanto en procesos de administración como de ingeniería de sistemas y software. Con su instauración se espera alcanzar beneficios como: calendarios y presupuestos predecibles en los proyectos, mejora del ciclo de vida dentro del desarrollo de software, mayor productividad, mayor calidad de los productos y servicios que ofrece la universidad a sus clientes y por ende la satisfacción de los mismos, además de mejorar la moral del personal que labora en el centro [26]. Para lograr las metas antes planteadas en el cumplimiento de las políticas establecidas, se definieron un conjunto de procesos relacionados con el ciclo de vida de los proyectos. Para dejar constancia de las actividades que se llevan a cabo se diseñó un expediente de proyecto compuesto por un conjunto de productos de trabajo que sirven para guardar información de los diferentes procesos en el desarrollo de software. Estos productos de trabajo inciden en los resultados para alcanzar el nivel de 2 CMMI

mediante la implementación de las áreas Administración de la configuración, Monitoreo y control de proyecto, Aseguramiento de la calidad del proceso y el producto, Administración de acuerdo con proveedores, Medición y análisis, Planificación del proyecto, y Administración de requisitos.

Teniendo en cuenta las características de los equipos de desarrollo de Nova y las exigencias de la organización se considera adecuada la utilización del expediente del programa de mejora en su versión 3.4, vinculándolo a la nueva configuración de la metodología OpenUp. Es importante en está decisión que dicho expediente tiene una estructura que permite dejar las evidencias necesarias para garantizar comunicación entre todos los involucrados, continuidad de los proyectos en el tiempo y apoyo para alcanzar el nivel 2 de CMMI. Es una desventaja la cantidad de plantillas propuestas, porque el proceso de documentación puede disminuir la velocidad del equipo de desarrollo. La solución puede estar en utilizar herramientas de apoyo a la gestión de proyectos como por ejemplo GESPRO y cuadernos de trabajo.

1.6 Roles responsables del desarrollo de las distribuciones de GNU/Linux Nova

Se estudiaron los roles propuestos en la metodología OpenUp, los analizados en los procesos de desarrollo de distribuciones de GNU/Linux y en el programa de mejora determinando que estos no son suficientes para desarrollar la distribución de GNU/Linux Nova. Después de realizar tormentas de ideas con los especialistas en el desarrollo de distribuciones GNU/Linux del departamento Sistemas Operativos quedó evidenciado que roles como el Mantenedor de paquetes, Administrador del kernel, Administrador de servicios telemáticos y el Coordinador de la metodología (existe en la metodología SCRUM) son necesarios para desarrollar la distribución de GNU/Linux Nova. Por tanto, se determina adicionar estos a los seleccionados del programa de mejora.

1.7 Infraestructura tecnológica del Departamento Sistemas Operativos para el desarrollo de distribuciones de GNU/Linux Nova

Para desarrollar la distribución de GNU/Linux Nova en el Departamento Sistemas Operativos se utilizan las herramientas:

GESPRO, herramienta de gestión de proyectos que permite: La planificación, el control y seguimiento de los proyectos y de los recursos asociados a los mismos. La planificación del alcance y el tiempo, la gestión de recursos humanos y sus competencias, la gestión de riesgos, así como la gestión financiera de los proyectos. El control y seguimiento de proyectos y un sistema para el diseño dinámico de reportes que permiten el acceso a la información del estado de los proyectos con diferentes niveles de detalle y la gestión documental con facilidades para la gestión del expediente de

los proyectos [27].

SITIO HUMANOS: Es un sitio web que permite la comunicación entre varios usuarios de la comunidad de Nova, permitiendo intercambiar experiencias sobre la utilización del sistema, reportar problemas, brindar soluciones o mejores procedimientos para ejecutar determinadas tareas sobre la distribución.

VISUAL PARADIGM: Es una herramienta CASE, que permite el modelado de software durante todo el ciclo de vida del proyecto, la generación de código fuente y la documentación de diferentes procesos.

REPREPO: Permite crear y gestionar repositorios.

APACHE: Permite montar servidores web, logrando que estos sean visibles para varios usuarios independientemente de su posición geográfica.

ZENTYAL: Es un servidor de red unificada de código abierto (o una plataforma de red unificada) para las pequeñas empresas. Zentyal puede actuar gestionando la infraestructura de red, como servidor de oficina, como servidor de comunicaciones unificadas o una combinación de estas. Además, incluye un marco de desarrollo para facilitar nuevos servicios basados en Unix [28].

GEANY: Es un editor de texto utilizando el toolkit GTK2 con las características básicas de un entorno de desarrollo integrado. Fue desarrollado para proporcionar un IDE (Entorno de Desarrollo Integrado) pequeño y rápido, que sólo tiene unas pocas dependencias de otros paquetes. [29]

PAQUETES BUILD ESSENTIAL: Conjunto de herramientas que permiten la compilación de paquetes de código.

PHORONIX TEST SUITE: Herramienta que permite hacer pruebas de rendimiento a distribuciones de GNU/Linux, comparando los resultados obtenidos con el de otras distribuciones.

GIT: Herramienta distribuida de control de versiones.

SUBVERSION: Herramienta centralizada de control de versiones.

Conclusiones del Capítulo 1

- La metodología más aceptable para desarrollar distribuciones GNU/Linux es OpenUp por brindar un ambiente configurable y extensible, permitiendo adicionar a la misma actividades y técnicas de desarrollo.
- Es común en varios proyectos que desarrollan distribuciones de GNU/Linux la utilización de herramientas colaborativas, estilos de dirección de proyecto tanto centralizados con descentralizados, la creación de proyectos de apoyo entorno a los desarrollos principales y el refuerzo de la documentación de procesos y aplicaciones.

- Los roles Mantenedor de paquetes, Administrador del kernel y Administrador de servicios telemáticos son importantes en el desarrollo de distribuciones de GNU/Linux.
- Según las características del equipo de desarrollo de la distribución de GNU/Linux Nova los productos de trabajo propuestos por OpenUp no son suficientes por lo que se determina vincular a la metodología antes mencionada una selección de los definidos en el programa de mejora.
- Para dar solución a la problemática se mantiene la misma infraestructura tecnológica que se utiliza en el Departamento Sistemas Operativos, esta ha demostrado ser suficiente en el desarrollo de la distribución GNU/Linux Nova.

CAPÍTULO 2 METODOLOGÍA PARA DESARROLLAR LA DISTRIBUCIÓN CUBANA GNU/LINUX NOVA.

Introducción

Este capítulo presenta la metodología Nova - OpenUp enfocada al desarrollo de la distribución GNU/Linux Nova. Nova - OpenUp es un híbrido que tiende hacia lo tradicional, mezcla buenas prácticas de las metodologías pesadas como la fuerte documentación, la profunda planificación, y de las ágiles el trabajo colaborativo, el intercambio constante entre desarrolladores y partes interesadas.

2.1 ¿Qué es Nova - OpenUp?

Nova – OpenUp es la unión y redefinición de un conjunto de buenas prácticas de la metodología de desarrollo de software OpenUp, de las políticas para alcanzar el nivel 2 de CMMI y de otras relacionadas con el desarrollo de distribuciones GNU/Linux en entornos de software libre. Incluye de la metodología SCRUM el rol coordinador de la metodología similar al SCRUM Master. Para aplicarla se debe tener como precondición que la misma se ajusta a proyectos de desarrollo de software. Es adecuada tanto para proyectos pequeños como grandes porque es configurable a las características del entorno donde se utilice. También se debe poseer una infraestructura tecnológica que permita la comunicación constante entre los desarrolladores de la distribución de GNU/Linux Nova y su comunidad de usuarios.

2.1.1 Componentes de la Metodología Nova - OpenUp

La siguiente figura muestra los componentes de la metodología Nova – OpenUp, en los siguientes epígrafes se explican los más importantes.

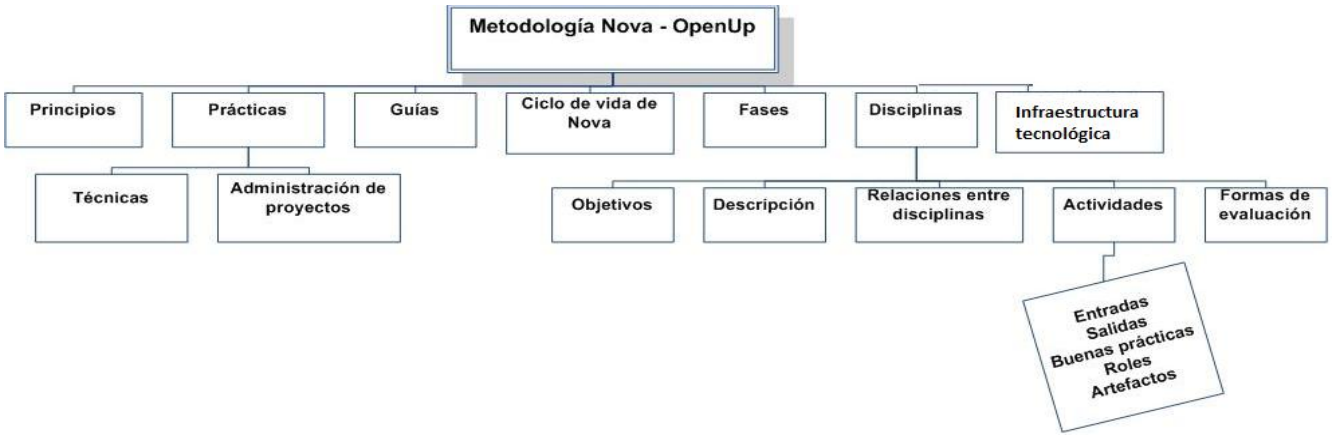


Figura 3 Componentes de la metodología

2.1.2 Principios de la Metodología Nova - OpenUp

Nova - OpenUp a los principios propuestos originalmente por OpenUp adiciona:

Utilizar una infraestructura tecnológica adecuada que fomente el trabajo en entornos de software libre, para maximizar la colaboración entre todos los involucrados, potenciando la utilización de herramientas colaborativas.

Aplicar métodos para la obtención de calidad en procesos y productos, teniendo en cuenta las políticas que define el programa de mejora emitido por CaliSoft en cuanto a las prácticas propuestas para el área de proceso de aseguramiento de la calidad (PPQA).

2.1.3 Prácticas de la Metodología Nova - OpenUp

Las prácticas en su mayoría son las mismas definidas por OpenUp, conservando las relacionadas con la administración de proyectos y las técnicas. Se sustituye en las prácticas técnicas:

Arquitectura evolutiva por Arquitectura Completa teniendo en cuenta que muchas veces esta es heredada de distribuciones ancestras.

Desarrollo guiado por casos de uso por Desarrollo guiado por requisitos, siendo esta última la técnica más adecuada para describir las funcionalidades de la distribución GNU/Linux Nova. Ésto teniendo en cuenta que se pueden describir requisitos como si fueran casos de prueba lo que va a aumentar la velocidad del desarrollo de software.

2.1.4 Fases de Nova - OpenUp

Nova – OpenUp traza un nuevo camino en la forma en que suceden las fases de desarrollo (Figura 4), porque en la fase de transición según las opiniones de terceros pueden surgir nuevas funcionalidades o la necesidad de modificar alguna. Estas sugerencias de cambio en los requisitos se pueden tratar de dos formas teniendo en cuenta el impacto que tengan en el desarrollo de software. Primero, pueden surgir requisitos para un nuevo proyecto y segundo, otros deben ser modificados o agregados en la fase de construcción. Las fases propuestas abarcan todo el proceso de desarrollo de la distribución de GNU/Linux Nova.

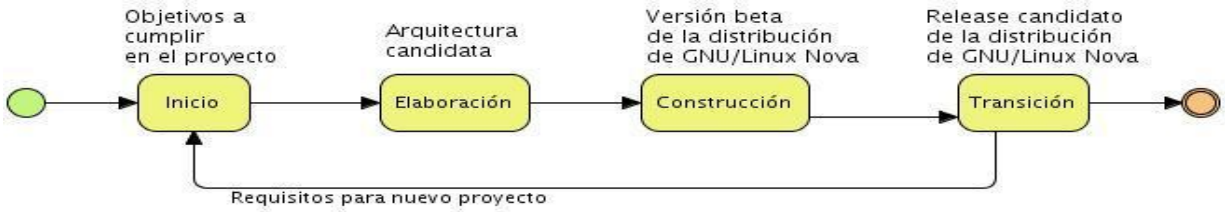


Figura 4 Fases de desarrollo de la distribución GNU/Linux Nova

En la **fase de inicio** se agrega el objetivo realizar un estudio del arte de las distribuciones similares determinando al menos una solución como base para el desarrollo del nuevo producto.

En la **fase de elaboración** se agregan los objetivos diseñar e implementar las diferentes vistas arquitectónicas desde las que se puede apreciar las distribuciones GNU/Linux y evaluar la

arquitectura propuesta determinando sus fortalezas y debilidades.

Los objetivos de la **fase de construcción** fueron redefinidos quedando como metas a cumplir: crear de forma iterativa una versión beta de la distribución GNU/Linux Nova y minimizar los costos de desarrollo mediante la optimización de recursos y la reutilización de componentes.

La nueva definición de los objetivos de la **fase de transición** plantea: obtener un release candidate de la distribución GNU/Linux Nova, validar el release candidate para determinar si cumple las expectativas de los usuarios, desplegar la distribución en entornos de los usuarios finales brindándoles servicios de transferencia de conocimientos y recolectar las opiniones que pueden convertirse en nuevos requisitos de software.

2.1.5 Roles

Los roles para la nueva metodología se obtienen de estudiar el Programa de mejora para alcanzar el nivel 2 de CMMi, las metodologías analizadas en el capítulo anterior, el resultado de tormentas de ideas con especialistas en el desarrollo de distribuciones GNU/Linux del departamento Sistemas Operativos y aspectos del desarrollo de las distribuciones Debian, Ubuntu y Fedora. Finalmente, se incorporan roles propuestos por el programa de mejora, además del mantenedor de paquetes, el administrador del kernel, el documentador, la comunidad de usuarios y el coordinador de la metodología. La figura 5 muestra que estos se dividen en dos niveles.

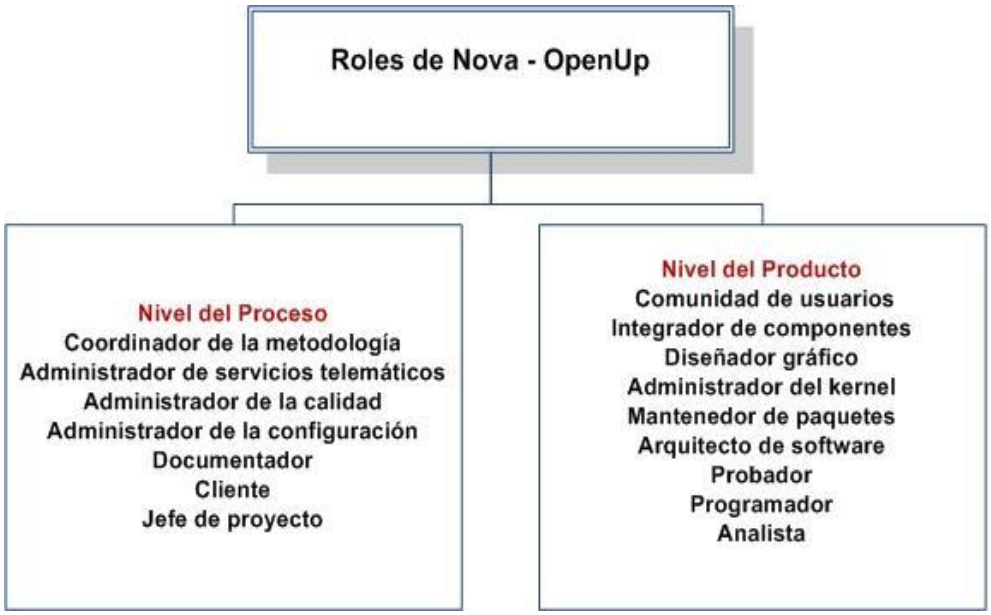


Figura 5 Roles para la metodología Nova - OpenUp

Las siguientes tablas muestran las habilidades y competencias de los roles que propone la metodología Nova – OpenUp.

Tabla 8: Roles pertenecientes al Nivel de producto [Fuente: Elaboración Propia]

Rol	Responsabilidades	Competencias
Comunidad de usuarios[6]	Revisa y corrige código fuente de las aplicaciones. Envía no conformidades. Programa requisitos. Envía nuevos requisitos. Crea manuales de aplicación.	Conocimiento de programación. Conocimiento de pruebas de software. Conocimiento de las funcionalidades del sistema. Conocimiento de herramientas colaborativas.
Integrador de componentes	Integrar componentes de software. Crear sistema instalable.	Conocimiento sobre estándares de programación, empaquetamiento y compilación de paquetes.
Diseñador gráfico	Realiza todo el diseño de las interfaces externas que requiere el sistema. Crea prototipos no funcionales. Define las pautas para el diseño de aplicaciones relacionadas con la distribución GNU/Linux Nova.	Creatividad. Dominio de técnicas de diseño. Habilidades de dibujo. Conocimientos y habilidades con las tecnologías de diseño. Comunicador. Tener iniciativa. Buen gusto. Objetividad [30]
Administrador del kernel	Documenta las modificaciones que van surgiendo en el kernel y las repercusiones de estas para el desarrollo que se lleve a cabo. Puede hacer modificaciones en el kernel.	Conocimientos referentes a los módulos, versiones de programas, compilaciones y arquitecturas del hardware y el software compatibles con el kernel del sistema operativo.
Mantenedor de paquetes	Corrige y actualiza paquetes disponibles en el repositorio. Empaqueta paquetes de código fuente. Compila los paquetes de acuerdo a los estándares establecidos. Sube nuevas versiones al repositorio de paquetes. Vela por la manera en que va cambiando un paquete determinado.	Conocimiento de técnicas de programación. Conocimiento sobre los paradigmas de la programación orientado a objetos y estructurada. Conocimiento sobre los lenguajes exigidos bash, python, perl, C//C++, JavaScript, XML. Conocimiento sobre herramientas de empaquetado. Conocimiento sobre herramientas de compilación de paquetes.
Arquitecto de software	Define todos los elementos bases de la arquitectura del proyecto. Identifica todos los posibles escenarios de despliegue de la aplicación. Identifica componentes de la aplicación. Determina las interfaces de integración tanto internas como externas. Elabora los documentos de arquitectura de software. Define las herramientas, bibliotecas, componentes, frameworks u otros componentes que permitan acelerar y mejorar el trabajo del proyecto. Define de conjunto con el jefe de proyecto el flujo de desarrollo basado en las	Dominio de patrones arquitectónicos. Dominio de patrones de diseño. Dominio de Hardware de servidores y computadoras personales. Dominio de herramientas de desarrollo y CASE. Dominio de despliegue de aplicaciones. Dominio de Estándares de información, código y comunicaciones. Dominio del diseño de redes de computadoras. Conocimientos profundos de la tecnología utilizada en el desarrollo [31]. Conocimiento sobre componentes de las distribuciones de GNU/Linux.

6 Esta parte de la comunidad de usuarios es la que agrupa a las personas que aspiran a convertirse en contribuidores de la solución de software y pueden ser probadores o desarrolladores de software.

Rol	Responsabilidades	Competencias
	herramientas identificadas. Vela por el cumplimiento de los requisitos de hardware. Responsable de la integración de los componentes del sistema.	
Probador	Elabora los planes de prueba. Da seguimiento a los planes de pruebas. Identifica los métodos, las técnicas, herramientas y directrices apropiadas para implementar las pruebas necesarias. Establece los escenarios de prueba en función de los requisitos. Dirige la definición del enfoque de prueba y garantiza la implementación satisfactoria. Ejecuta los casos de prueba y genera no conformidades asociadas a los mismos. Registra los resultados de las pruebas. Analiza los resultados de las pruebas realizadas y llega a conclusiones al respecto. Da seguimiento a la solución de las no conformidades detectadas. Evalúa el desempeño del probador y la calidad de las pruebas.	Habilidad en la lectura de planes y casos de prueba. Conocimiento de los enfoques y técnicas de las pruebas. Habilidades de diagnóstico y resolución de problemas. Conocimiento del sistema o aplicación que se somete a prueba. Formación en la utilización apropiada de las herramientas de automatización de prueba. Experiencia en la utilización de herramientas de automatización de prueba, especialmente en Phoronix Test Suite. Habilidades de programación. Habilidades de depuración y diagnóstico [31]
Programador	Convierte la especificación del sistema en código fuente ejecutable. Desarrolla el diseño teniendo en cuenta la arquitectura. Programa en función de los estándares de programación utilizados. Reutiliza código en la medida de lo necesario. Traduce códigos que pueden ser reutilizados de acuerdo a los lenguajes y estilos de programación exigidos. Elabora las pruebas unitarias y de integración. Desarrolla el prototipo de la interfaz de usuario. Integra los componentes que forman parte de la solución. Ejecuta los casos de prueba y genera no conformidades asociadas al mismo. Registra y analiza los resultados de las pruebas. Corrige las no conformidades relacionadas con la programación del sistema. Internacionaliza el código fuente de aplicaciones si es necesario.	Habilidades de comunicación. Conocimiento de técnicas de programación. Conocimiento sobre los paradigmas de la programación orientada a objetos y estructurada. Conocimiento sobre los lenguajes exigidos bash, python, perl, C/C++, JavaScript, XML. Conocimiento de análisis y diseño de sistemas. Habilidad sobre lógica y algoritmos. Conocimientos en procesamiento de datos. Aptitud para identificar la mejor alternativa de solución.
Analista	Participa con la comunidad de usuarios obteniendo las características relevantes que debe tener el sistema. Realiza el	Habilidades de comunicación. Capacidad de redacción y concreción. Habilidades de trabajo en equipo. Conocimiento de

Rol	Responsabilidades	Competencias
	estado del arte de las aplicaciones similares para determinar los requisitos. Captura los requisitos y define las prioridades. Realiza la especificación de requisitos. Realiza el seguimiento de los requisitos durante todo el desarrollo del proyecto. Participa en la definición de la arquitectura del sistema. Diseña las pruebas a realizar para verificar que se cumplieron los requisitos especificados. Documenta el estado del arte de las investigaciones en el expediente de proyecto exigido por la organización. Participa en la elaboración del Plan de Administración de Requisitos. Determina los proveedores válidos de requisitos. Crea y actualiza la Matriz de Trazabilidad. Crea manuales de ayuda a los usuarios en función de los requisitos funcionales del software.	metodologías de desarrollo de software. Conocimiento de ingeniería de software [30]. Conocimiento sobre la aplicación de técnicas de recopilación de la información, Conocimiento sobre la lógica y lenguajes de programación exigidos. Conocimiento sobre librerías básicas con las que se trabaja en el desarrollo de estos productos [28]. Conocimiento de metodología de la investigación científica.

Tabla 9: Roles pertenecientes al Nivel de proceso [Fuente: Elaboración Propia]

Rol	Responsabilidades	Habilidades
Cliente (Opcional)	Revisa y aprueba entregables del proyecto. Firma documentación legal del proyecto. Revisa el estado del proyecto.	Habilidades de comunicación. Capacidad de decisión. Conocimientos del negocio [30]
Coordinador de la metodología	Revisa que se trabaja de acuerdo a la configuración seleccionada de la metodología.	Conocimiento sobre la metodología Nova - OpenUp.
Administrador de los servicios telemáticos	Garantiza los servicios de redes del proyecto.	Dominio del diseño de redes de computadoras. Conocimientos sobre servicios telemáticos.
Administrador de la calidad	Elabora el Plan de Aseguramiento de la Calidad. Elabora el Plan de Mediciones. Participa en la elaboración del Plan de Monitoreo y en el monitoreo y análisis de las áreas de procesos. Participa en la elaboración de los planes de prueba. Participa en las revisiones técnicas formales de los artefactos. Participa en las revisiones de los entregables. Guía el diseño y ejecución de las pruebas internas. Participa en el análisis y recolección de los datos para las mediciones. Vela por el cumplimiento de	Trabajar en grupo. Ser buen comunicador. Ser líder. Dominar técnicas estadísticas. Poder de análisis estadístico. Dominar técnicas de recolección de información. Poder de síntesis. Ser persuasivo. Dominar el ciclo de desarrollo de software. Dominar las materias de ingeniería y gestión de software. Dominar la programación. Dominar el modelo CMMI. Explotar con efectividad herramientas de oficina. Dominar los tipos de pruebas. Dominar explotar herramientas de automatización de pruebas. Conocer los

Rol	Responsabilidades	Habilidades
	las políticas de la organización y reglas bases del proyecto. Colabora en las auditorías que se les realicen al proyecto. Coordina y colabora con las pruebas de liberación externa al proyecto. Crea una cultura de calidad en el proyecto. Participa en las revisiones de inconsistencias y las monitorea hasta su cierre.	principales estándares internacionales en la producción de software, así como los procedimientos y lineamientos que norma la producción en la UCI. Sensibilidad para detectar e identificar problemas. Aplicar técnica de dirección. Tomar decisiones. Conocer los principales conceptos relacionados con la calidad de software. Dominar los principios de gestión de la calidad. [31]
Documentador (Opcional)	Documenta el trabajo de los diferentes roles del proyecto, en el expediente exigido por la organización.	Conocimiento sobre el trabajo de los roles que va a documentar.
Planificador (Opcional)	Participa en la elaboración y actualización de los planes del proyecto. Elabora y controla cronogramas del proyecto. Planifica y gestiona los recursos del proyecto. Monitorea los planes del proyecto, cronograma y recursos.	Habilidades de comunicación. Ser organizado. Habilidades para trabajar en equipo [30]. Conocimiento sobre los requisitos de software que se deben cumplir. Conocimiento sobre los riesgos que se corren en el proyecto y la manera de mitigarlos. Conocimiento sobre herramientas de gestión de proyectos relacionadas con la planificación.
Jefe de proyecto	Participa en la fase de estudio preliminar (visión general del proyecto y análisis de factibilidad). Desarrolla el Plan de Desarrollo de Software. Aprueba las tecnologías a usar en el desarrollo del proyecto. Administra recursos. Participa en la legalización del proyecto. Realiza las estimaciones del proyecto. Define la organización del proyecto. Monitorea la adherencia a procesos. Participa en las Revisiones Técnicas Formales (RTF). Participa en las revisiones con la alta gerencia. Administra la capacitación interna al proyecto. Guía el proceso de identificación y mitigación de los riesgos. Evalúa a los miembros del proyecto según su desempeño. Genera y asigna acciones correctivas. Monitorea las acciones correctivas hasta su cierre. Es el responsable de determinar la necesidad de adquisición. Envía convocatoria a los proveedores. Selecciona y establece el acuerdo con el proveedor. Monitorea el acuerdo. Hace	Habilidades de liderazgo. Habilidades de comunicación. Conocimientos generales de las tecnologías utilizadas. Capacidad de decisión. Ser organizado. Habilidades para trabajar en equipo [30].

Rol	Responsabilidades	Habilidades
	pruebas de aceptación. Garantiza todos los recursos para la transferencia de la solución del proveedor al proyecto. Libera al proveedor del acuerdo.	
Administrador de la configuración	Identifica los elementos de configuración. Establece las líneas base del proyecto. Mantiene el control de las versiones de los artefactos del proyecto. Provee el mecanismo administrativo para precipitar, preparar, evaluar, aprobar o reprobar el procesamiento de propuestas de cambio. Genera métricas de la configuración del sistema. Configura y administra las herramientas para la administración de la configuración. Elabora Plan de Administración de la Configuración. Audita la gestión de configuración dentro del proyecto. Consulta y actualiza la Matriz de Trazabilidad.	Trabajar en grupo. Ser líder. Dominar el ciclo de desarrollo de software. Dominar las materias de ingeniería y gestión de software. Dominar la programación. Sensibilidad para detectar e identificar problemas. Conocer los principales conceptos relacionados con la calidad de software. Dominar los principios de gestión de la calidad. Organizado. Dominar la administración de requisitos. [30]

2.1.6 Productos de trabajo

La mayoría de los productos de trabajo que forman parte de la metodología se seleccionaron del expediente propuesto por el programa de mejora en su versión 3.4. Para su selección se tuvo en cuenta que sirvan de apoyo para alcanzar el nivel 2 de CMMI y guardar las evidencias necesarias del desarrollo de una distribución de GNU/Linux. Sobre los artefactos:

Se incorpora del programa de mejora versión 3.3 0213_Estado del Arte del producto a desarrollar.

De la disciplina Arquitectura se elimina 010216_1_Arquitectura_vista_de_procesos porque en el desarrollo de la distribución GNU/Linux Nova no se modelan los procesos del negocio y también se elimina 010216_4_Arquitectura_vista_de_datos y el diccionario de datos porque tampoco se trabaja con bases de datos. Se adiciona el artefacto Estándar de empaquetamiento.

A la disciplina Desarrollo se adicionan los artefactos Repositorio de paquetes, Paquete de código fuente, Paquete de código binario, Sistema Operativo Base (SOB[7]) y Sistema Instalable[8] (transita por las versiones Alfa, Beta y Realease Candidate).

7 Sistema Operativo Base GNU/Linux, es la combinación mínima necesaria de aplicaciones que permite la instalación de paquetes en un sistema GNU/Linux, que pueden o no soportar el arranque de la computadora. [7]

8 Sistema instalable: Es un archivo con extensión .iso que contiene un conjunto de aplicaciones compiladas según el hardware sobre el que va a funcionar, este debe estar listo para ser configurado en una memoria flash, en un CD o DVD, dispositivos que permiten la instalación del sistema en una máquina computadora, incluso sin disco duro en modo lifeCD. Este sistema instalable representa a la distribución de GNU/Linux Nova que tiene 3 variantes Nova Escritorio (para computadoras con hardware de altas prestaciones), Nova Ligero (para computadoras con hardware de bajas prestaciones) y Nova Servidores.

El artefacto 010114c_ Descripción_de_requisito_ágil va a tener dos propósitos, primero, guardar la descripción de los requisitos de la distribución y segundo servir como caso de prueba.

A la disciplina Prueba se adiciona el artefacto Plantilla de No Conformidades de la versión 2.2 del expediente de proyecto, porque esta permite gestionar las pruebas de manera fácil para el equipo de desarrollo.

Los productos de trabajo que se obtienen pueden tener una de las siguientes 3 clasificaciones Obligatorio, Opcional y Activado por Evento, estas definen la manera en que ellos van a ser utilizados en el proyecto.

2.1.7 Disciplinas de Nova – OpenUp

En la figura 6, se resalta el impacto que tiene cada una de las disciplinas en cada fase de desarrollo. Se pueden ejecutar varias iteraciones que emplean en mayor o menor medida actividades de las diferentes disciplinas y cada una resulta en un incremento, optimización o corrección del producto. Las cuatro primeras disciplinas mostradas tienen que ver con áreas técnicas como lo establece OpenUp. Las disciplinas dirección de proyectos y gestión de la calidad agrupan actividades tipo sombrillas, siendo añadida esta última a la metodología. Es interés de la autora de la investigación resaltar las actividades de aseguramiento de la calidad y otras de dirección de proyecto que resultan importantes para el desarrollo de la distribución GNU/Linux Nova.

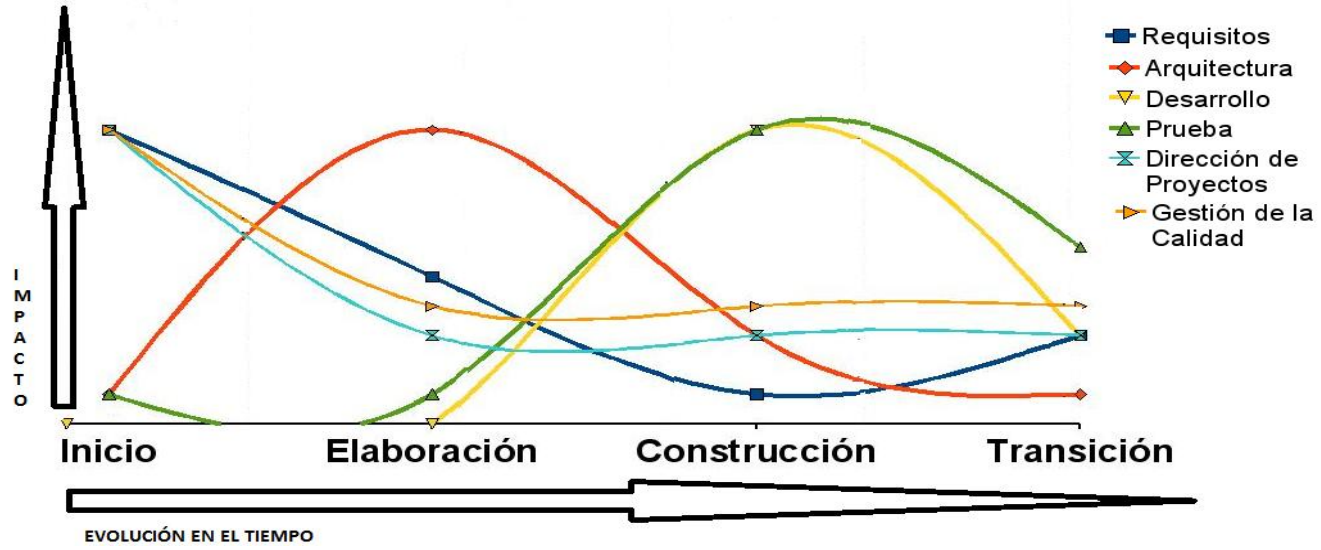


Figura 6 Impacto de las disciplinas en las fases de desarrollo de software

En las disciplinas se pueden encontrar actividades elementales para obtener una distribución GNU/Linux de manera ágil. Ellas simplifican al máximo los esfuerzos en cuanto a documentación del proceso y formación del personal. Las actividades Iniciar proyecto, Identificar requisitos, Negociar requisitos, Planificar proyecto, Crear entorno de trabajo, Detallar requisitos y Definir arquitectura

ocurren de manera secuencial y cada una de ellas pudiera constituir una iteración en dependencia de su complejidad. Una segunda iteración[9] puede agrupar las actividades: - Planificar Iteración, - Generar repositorio de binarios. Una iteración 3, puede agrupar las actividades: - Planificar Iteración, - Crear SOB y así sucesivamente hasta obtener un release candidate. De forma concurrente ocurren actividades de aseguramiento de la calidad encaminadas a controlar el proceso de desarrollo de software mediante la detección y solución de desviaciones a tiempo.

2.1.8 Relación entre disciplinas, actividades, roles y artefactos

Se utiliza la técnica de modelado de procesos BPM para hacer más entendibles las descripciones de las disciplinas. En los diagramas se presentan los principales roles, actividades y artefactos que pertenecen a una disciplina.

Disciplina Requisitos

Se agregaron actividades a la disciplina teniendo en cuenta el producto, el modelo CMMI y lo necesario para llevar a cabo un desarrollo comunitario.

Descripción: Esta disciplina proporciona las tareas necesarias para llevar a cabo la obtención, análisis, especificación, verificación, traceo, negociación, administración y validación de los requisitos de la distribución de GNU/Linux Nova.

Se redefine la actividad Identificar y refinar requisitos quedando dividida en las dos siguientes actividades:

Identificar requisitos de la distribución GNU/Linux (*)[10]: El propósito de esta tarea es capturar los requisitos funcionales y no funcionales de la distribución GNU/Linux. Se obtienen requisitos de los resultados de la fundamentación teórica de las distribuciones GNU/Linux que cumplan con los criterios de selección definidos, de las necesidades de los principales involucrados, de los reportes que se obtienen de los procesos de migración, de la opinión de los diferentes usuarios que pertenecen a la comunidad, de las exigencias de clientes específicos (si estos existen), de los paquetes de aplicaciones mínimas que necesita el SOB para su funcionamiento, de los reportes de los mantenedores de paquetes y del administrador del kernel, así como de la determinación del sistema anfitrión sobre el cual se va a desarrollar la distribución GNU/Linux.

Especificar requisitos de la distribución GNU/Linux (*): Permite especificar los requisitos de software siendo esta una herramienta para que todos los involucrados entiendan qué debe hacer la distribución a desarrollar y bajo qué condiciones.

Se eliminan el resto de las actividades mostradas en la tabla 5 de la fila Requisitos del Capítulo 1.

---

9 Iter: Iteración

10 **(*)**: Significa que la actividad es obligatoria, aún cuando se haga más ligera la metodología.

Se adicionan las actividades:

Detallar los requisitos de la distribución GNU/Linux (*): Esta tarea consiste en detallar suficientemente los requisitos funcionales, dejando claras sus características. Los requisitos deben ser descritos como si fueran casos de pruebas, lo que va a aumentar la velocidad de desarrollo.

Verificar los requisitos: Después que los requisitos son detallados es necesario evaluarlos para determinar si están descritos correctamente.

Negociar los requisitos (*): Determina la factibilidad, riesgos y costos de los requisitos, dicho análisis debe hacerse entre los involucrados relevantes, de dicha evaluación deben quedar compromisos formales para el cumplimiento de los requisitos.

Validar los requisitos (*): Comprueba que los requisitos del sistema estén correctos de acuerdo a las necesidades que debe satisfacer el producto y a las expectativas de los principales involucrados.

Establecer trazabilidad de los requisitos (*): Teniendo en cuenta la especificación de requisitos se conoce su evolución en el tiempo. Esta actividad dura todo el ciclo de vida del software y permite establecer lazos hacia adelante y atrás con los artefactos y entregables que surjan de los requisitos de software.

La figura 7 muestra las relaciones entre los principales roles, artefactos y actividades de la disciplina Requisitos. El rol Terceros agrupa a Clientes (si existen), Comunidad de usuarios y/o Organizaciones evaluadoras de la calidad del proceso de desarrollo de software.

Disciplina Arquitectura

Esta disciplina no recibe muchos cambios respecto a la metodología original, se debe tener en cuenta reutilizar las características de la distribución ancestra o de la arquitectura base (representa lo común para todas las variantes de Nova) y a esta modificarle o añadirle lo necesario para cumplir con los requisitos definidos. Además en las decisiones arquitectónicas se debe tener en cuenta el criterio de los roles integrador de componentes y administrador del kernel.

Las vistas arquitectónicas son una selección de las propuestas por el programa de mejora, estas quedan evidenciadas en los artefactos:

- 010216_0_Arquitectura_de_software
- 010216_2_Arquitectura_vista_de_sistema
- 010216_3a_Arquitectura_vista_de_presentación
- 010216_5_Arquitectura_vista_de_integración
- 010216_6_Arquitectura_vista_de_entorno_de_desarrollo_tecnológico
- 010216_7_Arquitectura_vista_de_seguridad
- 010216_8_Arquitectura_vista_de_infraestructura
- 010216_9_Vista_de_despliegue

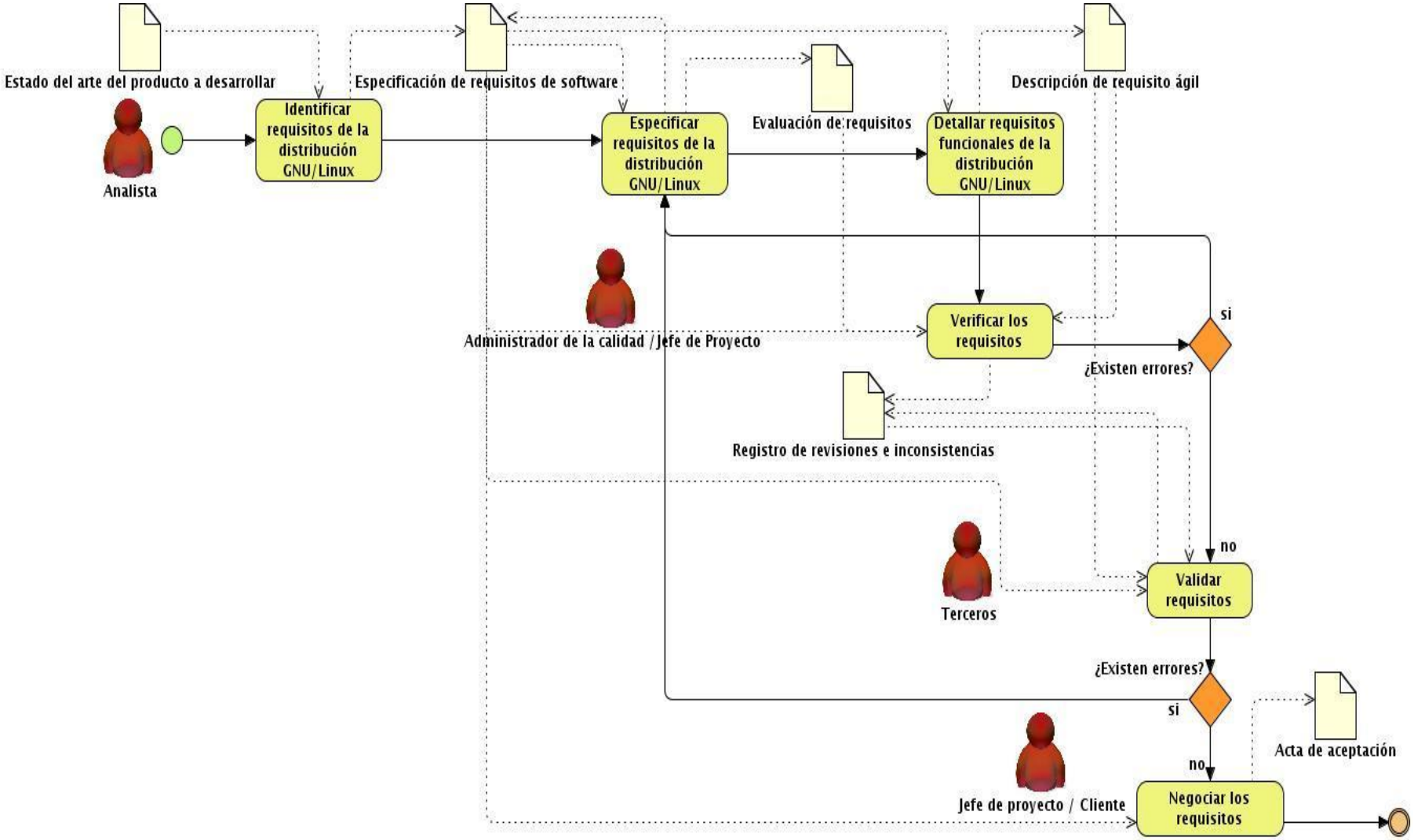


Figura 7 Relación entre las actividades, roles y artefactos de la disciplina Requisitos.

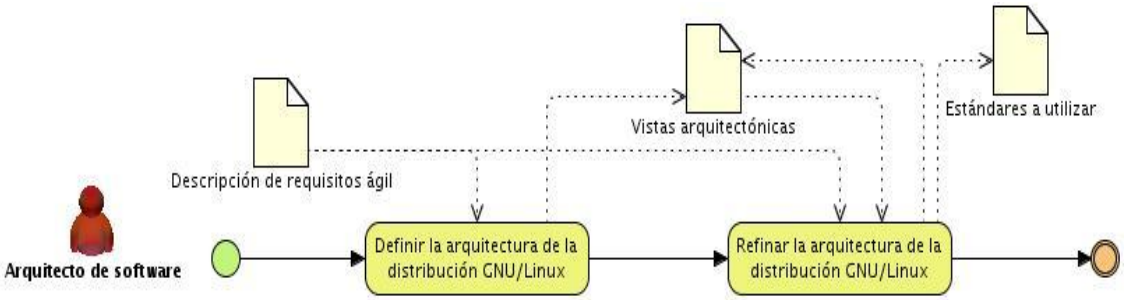


Figura 8 Relación entre las actividades, roles y artefactos de la disciplina Arquitectura

La figura 8 muestra la relación que existe entre las actividades, roles y artefactos contenidos en la disciplina Arquitectura.

Disciplina Desarrollo

Esta disciplina fue redefinida completamente respecto a la original pues se basa en actividades propias del desarrollo de distribuciones de GNU/Linux.

Descripción: Esta disciplina se encarga de implementar una solución técnica conforme a la arquitectura y a los requisitos de la distribución GNU/Linux. Se debe tener en cuenta que una distribución GNU/Linux está compuesta por un repositorio de paquetes de binarios y/o fuentes que contiene las aplicaciones a gestionar por el usuario y una imagen instalable del sistema operativo.

Relación con otras disciplinas

Requisitos: Definen qué hay que implementar y bajo qué restricciones.

Arquitectura: Es la disciplina que organiza y define las limitaciones de la implementación.

Pruebas: Revisan las diferentes versiones obtenidas en las actividades de desarrollo de la distribución.

Dirección de proyectos: Garantiza las planificaciones de las diferentes iteraciones a llevar a cabo.

Objetivos

1) Transformar los requisitos en funcionalidades y cualidades de la distribución.
2) Aplicar parches a los paquetes de código fuente que lo requieran.
3) Generar el repositorio de binarios.
4) Crear el sistema instalable.

Actividades

Generar repositorio de binarios (*): En un inicio para generar el repositorio de binarios es necesario descargar los repositorios de fuentes y binarios (si se hace de manera manual) que serán la base del repositorio a crear. Posteriormente, de manera cíclica a dicho repositorio se adicionan los paquetes de código binario que han sido actualizados de manera interna o externa al proyecto.

Crear parches a paquetes de código fuente que lo necesiten (*): Se descargan los paquetes de código fuente y los parches necesarios, se debe verificar si los paquetes necesitan mantenimiento (lo cual genera un parche) mediante la inspección minuciosa de su código.

Empaquetar paquete de código fuente (*): Cuando el código fuente está libre de errores puede ser empaquetado.

Compilar paquete de código fuente (*): Se revisa el paquete de código fuente y se compila en función de la arquitectura del hardware de la máquina. Dicha compilación muchas veces suele ser un proceso automatizado que termina con la instalación del código binario.

Crear sistema base (*): Para crear el sistema base se elabora una receta con los requisitos del sistema a crear, y a partir del repositorio y la receta al sistema base se le incorpora el núcleo del sistema operativo, el instalador y se empaquetan todos los componentes. El sistema base que se obtiene es para usuarios avanzados por lo que para ser utilizado por usuarios de conocimientos básicos se necesita hacer algunos retoques adicionando interfaces gráficas que potencien la comunicación entre los usuarios y el sistema.

Crear sistema instalable (*): A partir del sistema estándar que se creó en la actividad anterior que tiene lo elemental para el funcionamiento de la distribución, se crea un metapaquete al cual se le adiciona el entorno de escritorio y las aplicaciones requeridas, quedando conformada una imagen del sistema operativo instalable. En esta imagen debe estar incluida una ayuda que guié a los usuarios en la realización de las actividades que permite la distribución GNU/Linux.

Crear Manual de usuario (*): Para tener un producto completo se debe crear un manual de usuario que explique de manera clara y legible cómo se ejecutan las funcionalidades que forman parte de la distribución GNU/Linux. Siendo este un medio de instrucción sobre el sistema y los problemas que puedan suceder en la operación.

Crear Manual Técnico (*): Permite a usuarios administradores y desarrolladores de software administrar el sistema pudiendo así darle mantenimiento en caso que lo requiera.

Proceso: **Desarrollar software propio (*) (Opcional)**: Se pueden desarrollar nuevas aplicaciones para incorporarlas al repositorio de fuentes y binarios, para esto se debe iniciar un subproyecto / proyecto de desarrollo. En dependencia del producto a construir se define cómo va a ser su ciclo de vida y de qué manera se alinea con el de la distribución a obtener.

La figura 9 muestra las relaciones entre actividades, roles y artefactos de la disciplina Desarrollo.

Disciplina Pruebas

La disciplina Pruebas se redefine por completo. En la nueva actualización se tuvo en cuenta las experiencias adquiridas en el desarrollo de la distribución GNU/Linux Nova.

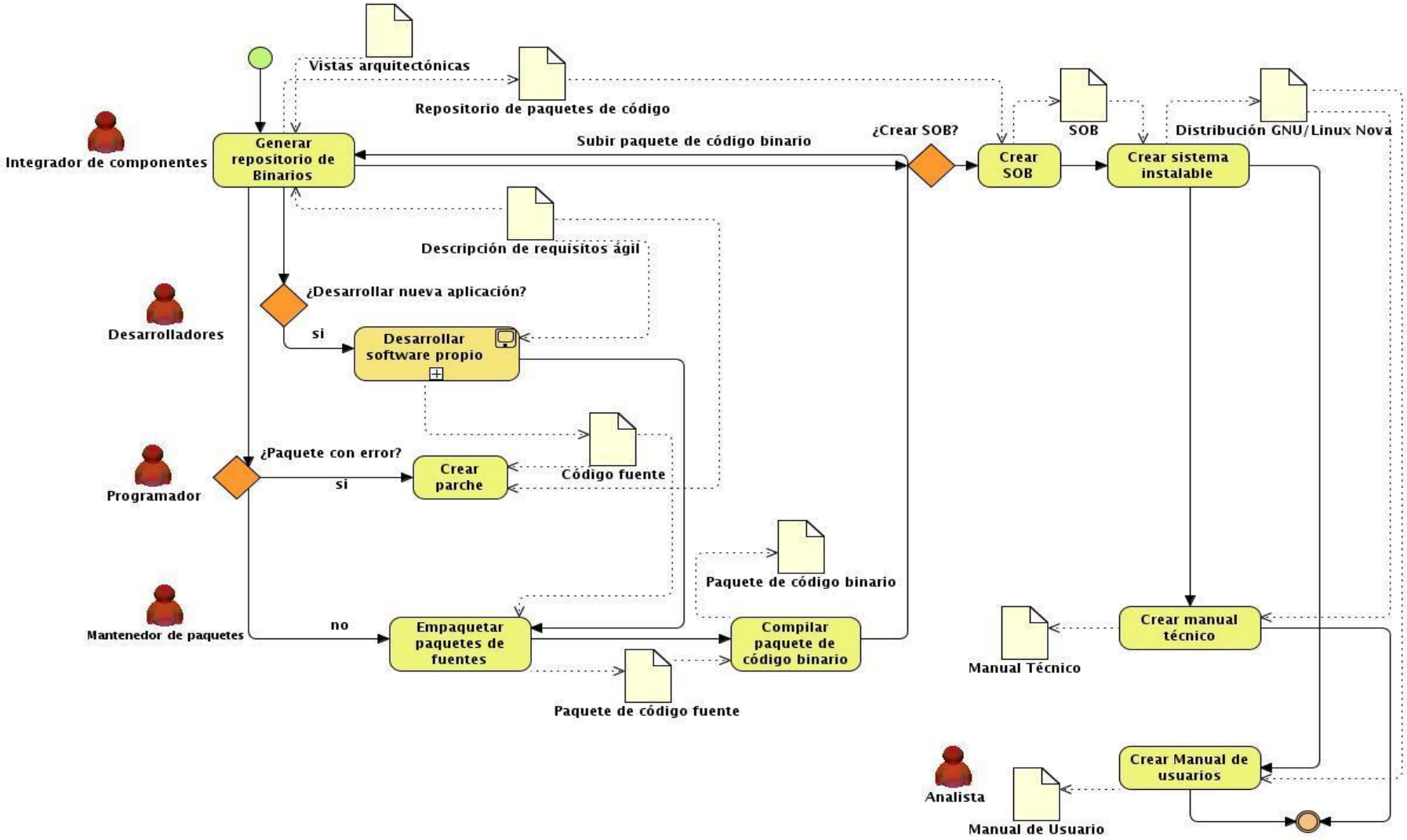


Figura 9 Relación entre las actividades, roles y artefactos de la disciplina Desarrollo

Descripción: Esta disciplina proporciona las tareas para llevar a cabo la evaluación de las distribuciones GNU/Linux, a través del diseño y ejecución de pruebas en función de los factores de calidad identificados.

Relación con otras disciplinas

Requisitos: Determina los objetivos que debe cumplir la distribución y con las pruebas se mide hasta que punto se cumplieron estos objetivos.

Desarrollo: En el desarrollo de las distribuciones GNU/Linux, cada paquete o módulo que se integre debe ser probado permitiendo hacer continuas correcciones al producto en función de los errores detectados.

Dirección de proyectos: Los resultados que aportan las pruebas permiten medir si ha cumplido con lo planificado la dirección de proyectos. Planifica los diferentes ciclos de pruebas a llevar a cabo [6].

Objetivos

1) Proporcionar información temprana y frecuente sobre el cumplimiento de los requisitos de la distribución.
2) Medir objetivamente el progreso durante la construcción de la imagen.
3) Asegurar que los cambios en el producto no van a introducir nuevos defectos.
4) Mejorar la velocidad del desarrollo de la distribución de GNU/Linux Nova.

Actividades

Diseñar casos de pruebas: El propósito de esta tarea es comprender los objetivos de la distribución GNU/Linux a desarrollar y las condiciones sobre las que debe ejecutarse, como resultado quedan definidas cuales van a ser las pruebas a realizar y los pasos a seguir para su ejecución. Parte de esta actividad se va a realizar mediante la descripción de requisitos como si fueran casos de prueba.

Probar arquitectura de software: Si la arquitectura ancestra ha recibido muchos cambios se recomienda hacer pruebas que muestren cuáles son los riesgos que pueden ocurrir de utilizarla y tomar decisiones para mitigarlos.

Probar internamente los paquetes (*): Existen herramientas de compilación que traen consigo un banco de pruebas que permiten hacer pruebas de caja blanca a los diferentes paquetes de código fuente antes de ser compilados, a esto se le suma que a los paquetes de código binario se le pueden hacer otras pruebas para ver si funcionan correctamente antes de subirlos al repositorio.

Implementar pruebas manuales (*): El propósito principal de esta tarea es verificar la calidad del producto y comunicarlo al equipo de desarrollo para que tome decisiones al respecto. Para ello se ejecutan los casos de pruebas, se analizan los resultados, se articulan los problemas, y se comunica los resultados al equipo.

Implementar pruebas automáticas (*): El objetivo fundamental de esta tarea es implementar o configurar las pruebas que se van a realizar de forma automática. Para lograrlo se deben seleccionar los requisitos que se van a medir, revisar la batería de pruebas que ofrece la herramienta utilizada, seleccionar las necesarias e implementar las pruebas que satisfagan el resto de los requisitos a medir automáticamente.

Poner distribución GNU/Linux a disposición de la comunidad de usuarios (*): En la medida que se cumplan las iteraciones se debe poner las versiones que se obtienen a disposición de la comunidad de usuarios, con la intención de que ellos se vayan relacionado con el producto antes de su culminación, obteniendo sus puntos de vista para el desarrollo, detección y corrección de errores.

Implementar pruebas de liberación por terceros (*): Es una buena práctica que pruebas que se hagan a los productos sean hechas por alguna empresa especializada, que tenga permisos para otorgar certificaciones que avalen que el producto desarrollado está listo para ser utilizado en el entorno de los usuarios finales.

La figura 10 muestra las relaciones entre los principales componentes de la disciplina Pruebas.

Disciplina Dirección de proyectos

Esta disciplina es modificada completamente porque se adicionan actividades de las áreas de planificación de proyectos y acuerdo con proveedores propuestas para alcanzar el nivel 2 de CMMI. Se recomiendan buenas prácticas resultado de experiencias del equipo de desarrollo de Nova. Como método de estimación de tiempo y de recursos se sugieren el tablón de Kanban asociado a una modificación de la estimación por puntos de historias de usuarios, juicio de expertos y analogías.

Descripción: Esta disciplina es la encargada de garantizar el correcto funcionamiento del equipo de trabajo desde su formación, planificación, hasta la garantía de los recursos necesarios para que puedan ejecutar sus tareas mediante la determinación de los riesgos que pueden atentar contra el desarrollo de software y la creación de un entorno de trabajo adecuado.

Relación con otras disciplinas

La dirección de proyectos envuelve actividades de tipo sombrilla que sirven de apoyo a todas las disciplinas para desarrollar software, garantiza que a cada actividad se le asigne un espacio para que pueda suceder.

Objetivos

1) Priorizar la secuencia de trabajo en función de los principales involucrados.
2) Estimular a un trabajo creativo y colaborativo a partir de las planificaciones del proyecto.
3) Monitorear constantemente las tareas planificadas.
4) Crear un ambiente de trabajo efectivo maximizando la productividad del equipo.

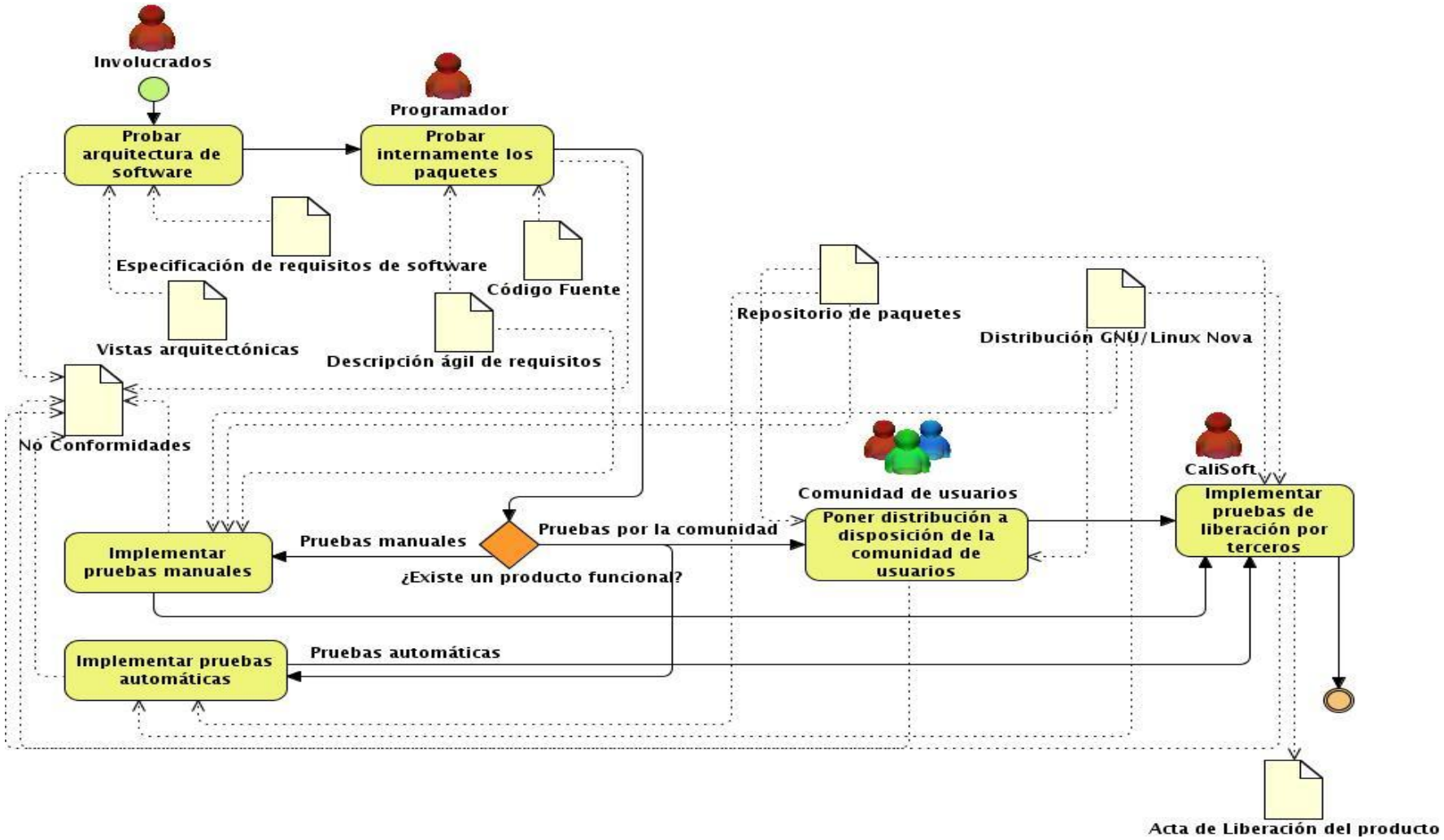


Figura 10 Relación entre las actividades, roles y artefactos de la disciplina Pruebas

5) Crear los mecanismos que permiten la continua información del progreso del proyecto a todos los involucrados.
6) Proveer un marco de trabajo que permita la gestión de riesgos y la adaptación a los cambios.

Actividades

Iniciar proyecto (*): En el inicio del proyecto se debe hacer un estudio de viabilidad en el que se detecten oportunidades de acuerdo a los propósitos que se quieren alcanzar, evaluar alternativas existentes para escoger la mejor, detectar y analizar los problemas que llevan al surgimiento del proyecto. Las actividades antes mencionadas brindan la información suficiente para determinar si se cuenta con el presupuesto y recursos necesarios para emprender dicho proyecto.

Planificar el proyecto (*): Esta actividad se encarga de definir qué hacer y cuándo para lograr que el proyecto logre sus objetivos. En esta planificación se debe analizar las necesidades de la comunidad de usuarios en cuanto a tiempo y prioridades. Las propuestas hechas son debatidas y si es necesario modificadas por el equipo de desarrollo de software, que debe comprometerse a cumplir con las fechas y metas escogidas. Las planificaciones del proyecto deben ser actualizadas a medida que este progresa teniendo en cuenta el resultado de la retroalimentación.

El paso previo a la planificación es la estimación del proyecto en cuanto a esfuerzo, alcance y costo. Existen muchos métodos que permiten hacer las estimaciones necesarias, los más utilizados en entornos de software libre son Juicio de expertos y Analogías pues logran incorporar al desarrollo actual experiencias existentes de entornos similares. Un método muy apropiado para el desarrollo de la distribución GNU/Linux es la utilización del tablón de Kanban por el cual deben transitar todos los requisitos a cumplir ordenados por prioridad. Esta prioridad la adquieren según su complejidad, tamaño, riesgos y disponibilidad de recursos. Estos transitan el tablón según las iteraciones a la que son asignados. La situación antes descrita aumenta la velocidad del equipo de desarrollo y la curva de aprendizaje del jefe de proyecto y/o planificador.

Planificar las actividades de aseguramiento de la calidad del producto y del proceso: En función del producto y el cronograma del proyecto se deben planificar las actividades de aseguramiento de la calidad, teniendo en cuenta el control constante del desarrollo y la retroalimentación continua.

Ejecutar las planificaciones: Se deben ejecutar todas las actividades definidas en el plan de desarrollo de software, así como las reglas establecidas para el correcto funcionamiento del proyecto.

Establecer acuerdos con proveedores (*) (Opcional): Seleccionar los proveedores de acuerdo a las necesidades del proyecto, dichos proveedores pueden brindar servicios, productos e incluso recursos humanos y materiales.

Crear entorno de trabajo adecuado (*): En función de los requisitos a cumplir se debe crear un entorno de trabajo adecuado que favorezca el trabajo colaborativo entre los desarrolladores y la comunidad de usuarios, además de la formación en algunos temas tecnológicos si es necesario. A esto sumarle las acciones necesarias para lograr comenzar el proyecto dentro del presupuesto establecido, con los recursos humanos y materiales adecuados. En los entornos de desarrollo de distribuciones de GNU/Linux es muy importante garantizar herramientas colaborativas que permitan gestionar el proyecto de manera tanto centralizada como descentralizada y crear otros proyectos entorno al desarrollo principal que apoyen la documentación, divulgación, traducción, etc.

Formar al personal involucrado en las actividades de desarrollo de la distribución GNU/Linux: Se deben crear los medios que permitan la formación del nuevo personal en el desarrollo de distribuciones GNU/Linux, utilizando como guía principal las competencias y habilidades que debe desarrollar cada rol.

Cerrar proyecto (*): Permite finalizar todas las actividades de las disciplinas del proyecto cerrando los contratos, actividades o temas pendientes. Antes de entregar el producto a la Comunidad de usuarios se les debe garantizar un conjunto de acciones de soporte y mantenimiento, asegurando material que permita instrucción sobre cómo utilizar la distribución GNU/Linux y la garantía de los canales de comunicación para la corrección de errores que aparezcan del lado de los usuarios. Finaliza el proyecto con la confección del acta de terminación del proyecto y la posterior inscripción de los productos desarrollados.

La figura 11 relaciona los principales roles, actividades y artefactos propuestos por la metodología.

En la planificación de proyectos intervienen varios roles, por Involucrados se entiende a los administradores de la calidad y la configuración, Coordinador de la metodología, Arquitecto de software y Analista.

Disciplina Gestión de la Calidad de Software

Se adiciona a la metodología esta disciplina por la importancia que tienen los procesos de calidad de software en la disminución de los costos relacionados con la corrección de errores y la satisfacción de los usuarios finales.

Descripción: El propósito principal de está disciplina es proveer los métodos necesarios para garantizar la calidad de software, teniendo en cuenta la corrección a tiempo de los errores, la uniformidad del trabajo, el control de los cambios y la mejora continua.

Relación con otras disciplinas: Esta es otra de las disciplinas que se compone de actividades sombrillas, dichas actividades sirven de apoyo a las ingenieriles garantizando que sucedan adecuadamente.

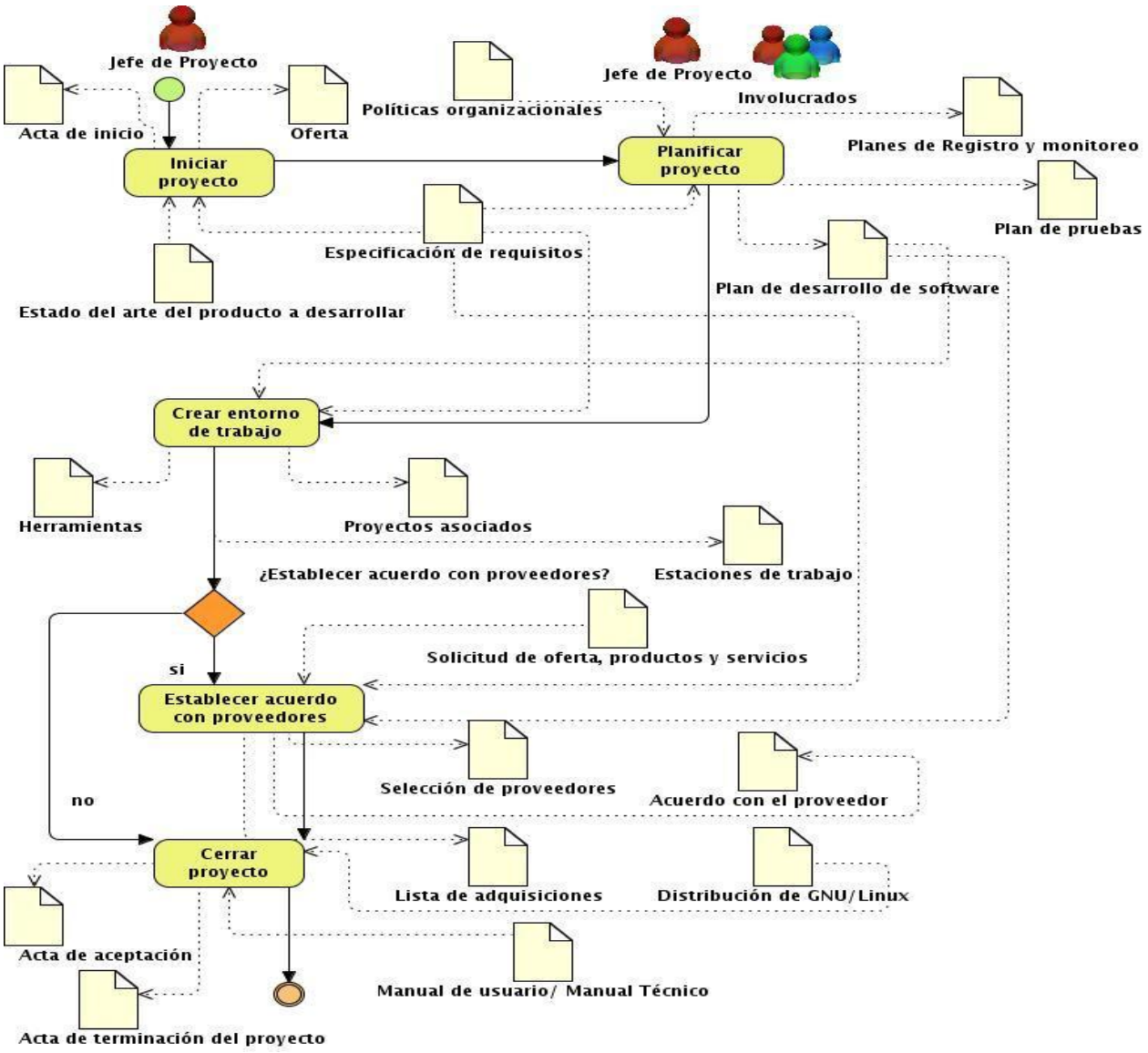


Figura 11 Relación entre las actividades, roles y artefactos de la disciplina Dirección de Proyectos

Objetivos

1) Proveer métodos que permitan la capacitación de los involucrados en las actividades de aseguramiento de la calidad.
2) Apoyar la obtención de calidad en los procesos de desarrollo de las distribuciones GNU/Linux.
3) Medir los procesos de desarrollo de software.
4) Controlar, evaluar y monitorear la evolución de los componentes de las distribuciones GNU/Linux.

5) Velar por el cumplimiento de las políticas emitidas por la organización.
6) Garantizar la mejora continua de los procesos que se ejecutan en el desarrollo de software.
7) Asegurar la correcta gestión de la configuración de software propiciando la adecuada evolución del producto.

Actividades

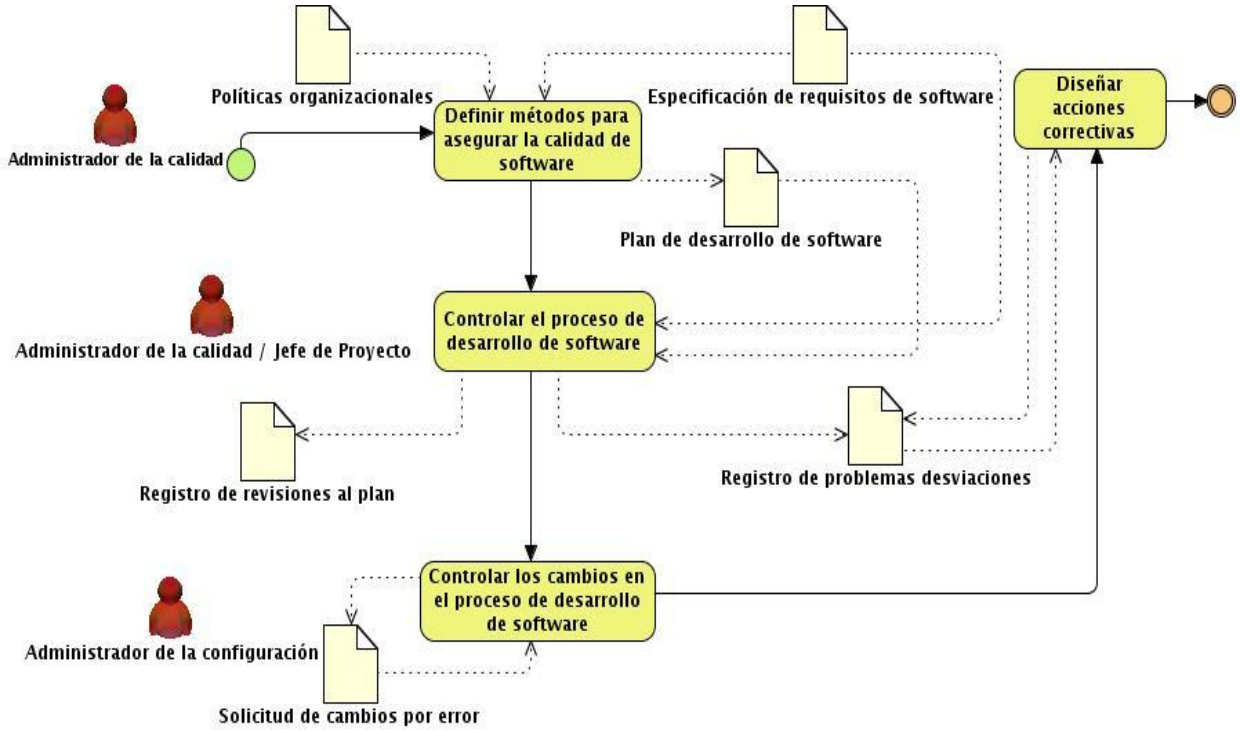


Figura 12 Relación entre las actividades, roles y artefactos de la disciplina Gestión de la Calidad

Definir métodos para asegurar la calidad del producto: En función del desarrollo de las distribuciones GNU/Linux y de los requisitos a cumplir se deben definir los métodos para asegurar la calidad del software. Existen normas, modelos, estándares, técnicas de medición del proceso que pueden ser utilizadas o adaptadas en función de las necesidades identificadas.

Controlar el proceso de desarrollo de software: El control del proceso de desarrollo de software esta compuesto de actividades de monitoreo que permiten detectar a tiempo las desviaciones que pueden surgir y corregirlas. Las actividades de control de la calidad pueden tener menor o mayor complejidad en dependencia de los objetivos de las mismas y pueden ser revisiones, inspecciones, auditorías y/o la aplicación de métricas a los diferentes elementos del proyecto.

Diseñar acciones correctivas: El diseño de acciones correctivas propone soluciones adecuadas a las desviaciones que se dan en el proceso de desarrollo de software, estas son analizadas entre los

involucrados y perfeccionadas de modo que puedan ser implementadas para eliminar los problemas detectados.

Controlar los cambios en el desarrollo de software (*): Esta actividad se encarga de crear los mecanismos para controlar los cambios que suceden en el proyecto. Para lograr esta tarea con éxito se deben seguir los siguientes pasos: identificar los elementos de la configuración de software, establecer las líneas bases, definir mecanismos para controlar los cambios que ocurren a diferentes niveles y auditar el estado de la configuración de software.

Las disciplinas actuales definen de manera muy superficial las actividades de despliegue y soporte porque estas no son responsabilidad del equipo de desarrollo de la distribución GNU/Linux Nova, de igual manera el proceso que se explica es para el desarrollo de una nueva distribución a partir de otra ya existente.

La figura 12 muestra las principales relaciones entre las actividades, roles y artefactos.

Conclusiones del Capítulo 2

En el actual capítulo se llega a las siguientes conclusiones sobre la metodología Nova - OpenUp:

- Es el resultado de unir una selección de buenas prácticas de las metodologías OpenUp, Scrum, Kanban, el Programa de Mejora para optar por el nivel 2 de CMMI y las necesidades del proceso de desarrollo para crear las variantes de la distribución de GNU/Linux Nova.
- Está compuesta por principios, prácticas, fases, disciplinas, actividades, buenas prácticas, roles, artefactos, listas de chequeo y sus relaciones, elementos que sirven de guía para obtener la distribución de GNU/Linux Nova.
- Es una metodología que responde al enfoque tradicional, sin embargo puede ser más ligera mediante la sustracción de actividades.

CAPÍTULO 3 EVALUACIÓN DE LA METODOLOGÍA PARA EL DESARROLLO DE LA DISTRIBUCIÓN DE GNU/LINUX NOVA

Introducción

Para evaluar si la metodología es correcta se utiliza el método Juicio de expertos. Se implanta la metodología parcialmente en un ambiente real de desarrollo de distribuciones GNU/Linux pudiendo valorar la situación antes y después de la existencia de la misma. Por último, se compara la metodología con reconocidos estándares y se determinan sus fortalezas y debilidades.

3.1 Evaluación de la variable independiente Metodología para desarrollar la distribución de GNU/Linux Nova

Para evaluar esta variable se utiliza el método Juicio de expertos, haciéndole una adaptación teniendo en cuenta que el entorno de trabajo seleccionado no posee expertos en el tema sino especialistas de gran experiencia.

Objetivo a alcanzar

Evaluar la metodología de desarrollo de distribuciones de GNU/ Linux Nova teniendo en cuenta los indicadores:

- Estado correcto de la metodología.
- Nivel de importancia de la metodología.
- Nivel de suficiencia de la metodología.

3.1.1 Selección de especialistas

Para seleccionar a los especialistas se tuvo en cuenta el conocimiento sobre una o varias de las siguientes áreas: desarrollo de distribuciones de GNU/Linux, ingeniería de software y calidad de software, dentro de la muestra se encuentran especialistas de los Departamentos Sistemas Operativos, el Central de Ingeniería y Gestión de Software y de la Dirección de Calidad. Se escogieron 20 especialistas, de ellos solo se obtuvo confirmación para participar en la investigación de 13.

Para determinar el coeficiente de competencia de los especialistas se utiliza la fórmula:

K = ½ (Kc + Ka) donde:

K = Coeficiente de competencia, **Kc** = Coeficiente de conocimientos y **Ka** = Coeficiente de argumentación

Para determinar Kc se les orientó a los especialistas que valoraran el nivel de conocimientos que poseen en un rango del 0 al 10, teniendo en cuenta que 0 es ausencia de conocimientos y 10 es el máximo nivel de conocimientos, los resultados obtenidos se presentan en la siguiente tabla:

Tabla 10: Valoración del nivel de conocimientos por los especialistas. [Fuente: Elaboración Propia]

Expertos (E)	0	1	2	3	4	5	6	7	8	9	10
E1							X				
E2							X				
E3							X				
E4								X			
E5								X			
E6								X			
E7						X					
E8						X					
E9							X				
E10							X				
E11							X				
E12					X						
E13									X		

La tabla 11 expresa el coeficiente de conocimientos de cada especialista mediante la formula **Kc = criterio x 0,1**

Tabla 11: Coeficiente de conocimientos de cada especialista. [Fuente: Elaboración Propia]

E	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13
Kc	0,6	0,6	0,6	0,7	0,7	0,7	0,5	0,5	0,6	0,6	0,6	0,4	0.8

Para obtener el Ka se hace uso de la siguiente tabla patrón.

Tabla 12: Tabla Patrón. [35]

Fuentes de argumentación	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted	0.3	0.2	0.1
Su experiencia obtenida	0.5	0.4	0.2
Trabajos de autores nacionales	0.05	0.05	0.05
Trabajos de autores extranjeros	0.05	0.05	0.05
Su propio conocimiento del estado del problema en el extranjero	0.05	0.05	0.05
Su intuición	0.05	0.05	0.05

La siguiente tabla muestra los resultados obtenidos para cada Ka según el especialista.

Tabla 13: Resultados obtenidos para cada Ka según el especialista. [Fuente: Elaboración Propia]

E	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13
Ka	1	1	0,9	1	0,9	0,9	0,9	0,6	0,9	1	1	0,9	0,9

Finalmente se obtiene el coeficiente de competencia de cada especialista.

Tabla 14: Coeficiente de competencia de cada especialista. [Fuente: Elaboración Propia]

E	E1	E2	E3	E4	E5	E6	**E7**	**E8**	E9	E10	E11	**E12**	E13
K	0,8	0,8	0,75	0,85	0,8	0,8	**0,7**	**0,55**	0,75	0,8	0,8	**0,65**	0,85
Nivel	Medio	Medio	Medio	Alto	Medio	Medio	**Medio**	**Medio**	Medio	Medio	Medio	**Medio**	Alto

Se escogieron los especialistas con Nivel Alto y Medio que tuvieran el K por encima de 0.7 para mayor calidad en la investigación, eliminando así los especialistas señalados en **negrita** en la tabla anterior. Los especialistas seleccionados tienen más de 6 años de experiencias en los temas de interés.

3.1.2 Indicador (I) 1 Estado correcto de la metodología

Para evaluar si la metodología es correcta se utiliza el Modelo de evaluación de metodologías para el desarrollo de software propuesto por MÉNDEZ NAVA Elvia Margarita, para optar por el título de especialista en gestión de proyectos de la Universidad Católica Andrés Bello.

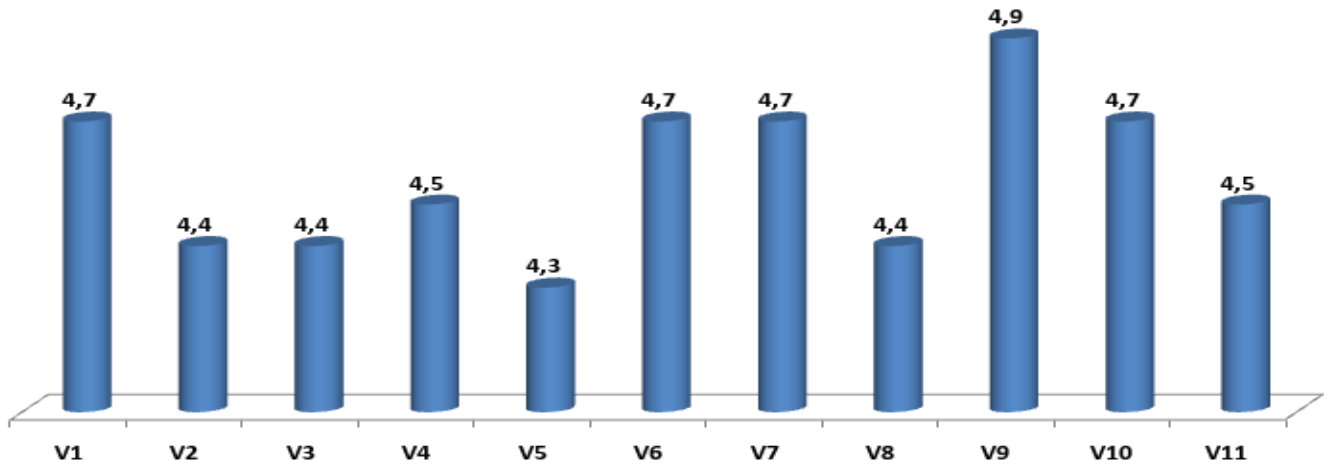


Figura 13 Resumen de la evaluación de la metodología según modelo de la Universidad Andrés Bello

Se confecciona un cuestionario (Ver Anexo 3) basado en las variables (V) definidas en el modelo y en las características claves que debe cumplir una metodología para adecuarse al desarrollo de distribuciones de GNU/Linux.

Variable 1: La metodología debe ajustarse a los objetivos.

Variable 2: La metodología debe soportar la determinación de la exactitud del sistema a través del ciclo de desarrollo.

Variable 3: La metodología debe ser la base de una comunicación efectiva.

Variable 4: La metodología debe funcionar en un entorno dinámico orientado al usuario.

Variable 5: La metodología debe especificar claramente los responsables de los resultados.

Variable 6: La metodología se debe de poder enseñar.

Variable 7: La metodología debe estar soportada por herramientas CASE.

Variable 8: La metodología debe soportar la eventual evolución del sistema.

Variable 9: La metodología debe contener actividades conducentes para mejorar el proceso de desarrollo de software.

Variable 10: La metodología debe contener actividades para el trabajo en entornos de software libre.

Variable 11: La metodología debe integrar procesos del desarrollo de distribuciones de GNU/Linux.

Los especialistas otorgaron una nota de 0 a 5 a la metodología según la presencia correcta de cada variable, sabiendo que 0 es la inexistencia de la variable en la metodología y 5 es el valor máximo que puede tener una variable si existe de manera correcta. La figura 13 muestra el resumen de los valores otorgados, se puede observar que el promedio de notas más bajo es 4.3, lo que demuestra que según el modelo utilizado la metodología es bastante correcta.

El 90% de los especialistas dice que la metodología es correcta (Figura 14) pues está basada en las prácticas propuestas por especialistas en el desarrollo de distribuciones GNU/Linux, las sugeridas de ingeniería de software y por el modelo CMMI. Solo existe un especialista en desacuerdo que expresa que para que llegue a ser completamente correcta necesita ser utilizada en varios ambientes reales de desarrollo de distribuciones GNU/Linux.

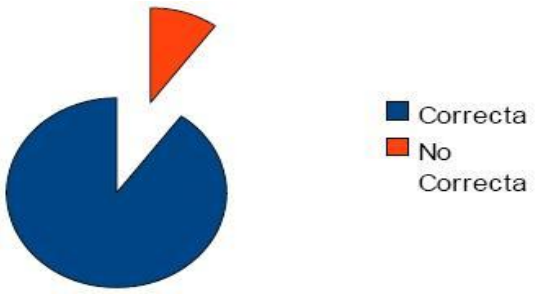


Figura 14 Carácter de corrección de la metodología.

3.1.3 Indicador Nivel de importancia de la metodología

Las respuestas al cuestionario aplicado (Ver Anexo 3), plantean que este tipo de desarrollo

tradicionalmente se lleva a cabo de forma anárquica, lo que trae como consecuencia que se trabaje sin tener los objetivos correctamente definidos. Los desarrolladores crean los programas sin guiarse por un diseño y sin documentar el proceso, además se duplican constantemente los esfuerzos. El 100% de los especialistas concuerda que la metodología es importante (Figura 15), porque permite obtener distribuciones GNU/Linux de forma organizada y fácil, lo que acelera el proceso de desarrollo mediante la creación de un ambiente adecuado, alineado a las políticas de la organización e influyendo positivamente en la calidad del producto.

Figura 15 Importancia de la metodología.

3.1.4 Indicador Nivel de suficiencia de la metodología

El 70% de los especialistas plantea que la metodología es suficiente (Figura 16) pues está compuesta por todas las actividades necesarias para obtener una distribución GNU/Linux explicando de forma detallada qué hacer, cómo, cuándo y quién es el responsable. La mayoría de los especialistas consideran que la metodología es completa porque contiene la definición de sus diferentes fases, disciplinas, actividades, roles, artefactos, así como sus relaciones. Opinan además, que no es compleja su implementación porque está definida en función de las necesidades reales de la distribución GNU/Linux Nova y es muy explicativa lo que facilita su aprendizaje. Un especialista considera que no es completa hasta que no posea un producto que la avale. Se expresa como desventaja el hecho de que para equipos inexpertos se necesitaría de un supervisor o especialista en ingeniería de software en los primeros tiempos de trabajo. El 10% de los especialistas recomienda fortalecer las actividades de soporte y sugerir qué infraestructura tecnológica sería la más adecuada para sustentar las prácticas que sugiere la metodología.

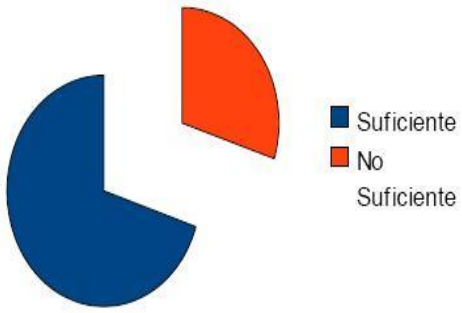


Figura 16 Suficiencia de la metodología.

Se puede observar en la figura 17 que de manera general los especialistas valoran positivamente la metodología para el desarrollo de la distribución GNU/Linux Nova siendo el menor porciento de aceptación 70, y este está por encima de la media.

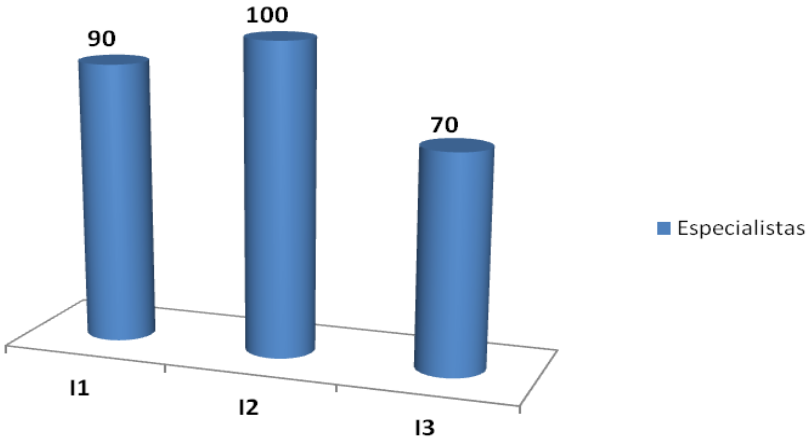


Figura 17 Resumen de la valoración de la variable independiente

3.2 Evaluación de la variable dependiente Desviaciones existentes en el proceso de desarrollo de la distribución GNU/Linux Nova

El método experimental permite de forma sistemática observar eventos, coleccionando datos, analizando información y reportando resultados. Para aplicar el método experimental se valora la situación antes y después de implementar la metodología Nova - OpenUp llegando a criterios positivos y negativos de los cambios que suceden. Para evaluar la metodología se plantea como **hipótesis nula**:

Ho: Si se utiliza una metodología para desarrollar la distribución de GNU/Linux Nova, caracterizada por estar integrada al programa de mejora propuesto por CaliSoft en su versión 3.4, contener actividades básicas del trabajo en entornos de software libre e integrar procesos claves del desarrollo de distribuciones de GNU/Linux, esta no contribuirá a la disminución de las desviaciones existentes

en el proceso de desarrollo de la distribución GNU/Linux Nova en la UCI.

Para determinar la situación antes de existir la metodología se tuvieron en cuenta los resultados de la auditoría realizada en noviembre de 2010 a los proyectos Nova Base, Nova Escritorio, Nova Servidores y Nova Ligero por parte del proyecto Nova QALIT. Posteriormente, como parte de las acciones correctivas a las desviaciones detectadas se diseñó e implantó parcialmente la metodología Nova – OpenUp en estos proyectos. Los resultados posteriores a la implantación de la metodología se obtienen de realizar auditorías trimestrales y revisiones realizadas por CaliSoft y el proyecto Nova QALIT. El proceso de asimilación de la nueva tecnología ha sido enriquecedor teniendo en cuenta que ha permitido modificaciones en la metodología y en el proceso de desarrollo de la distribución de GNU/Linux Nova, esto ha influido en la corrección y prevención de errores a tiempo. Se miden los indicadores según las unidades de medidas especificadas en la tabla de operacionalización de variables (Introducción de la Tesis).

3.2.1 Indicador 1 Grado de utilización de una metodología de desarrollo de software

Tabla 15: Grado de utilización de la metodología [Fuente: Elaboración Propia]

Antes		***Después***	
Medio 50%	Los proyectos Nova Ligero, Nova Base, Nova Escritorio y Nova Servidores utilizaban la metodología SXP parcialmente debido a que muchas de las prácticas que esta propone no se ajustaban al desarrollo de Nova. Esto provocó que se obviaran pasos importantes y no se tuvieran las evidencias necesarias.	Alto 100%	Para evitar incongruencias se realizó un taller que tuvo como objetivo principal presentar la nueva metodología a los desarrolladores de la distribución GNU/Linux Nova, provocando así un debate para identificar sus fortalezas y debilidades. Esta actividad influyó positivamente en la metodología pues permitió la adición de nuevos artefactos que antes no se tuvieron en cuenta. Hoy, los 4 proyectos trabajan con la metodología Nova - OpenUp. Los integrantes tienen conocimiento de su funcionamiento pues se ven identificados con las actividades que propone, siendo estas representativas de su proceso de desarrollo actual (Ver Anexo 4).

3.2.2 Indicador 2 Nivel de adecuación de los roles que se dedican al desarrollo de distribuciones GNU/Linux

Tabla 16: Nivel de adecuación de los roles [Fuente: Elaboración Propia]

Antes		***Después***	
Medio 68,7%	Existía gran variedad de roles, muchos de ellos con las mismas responsabilidades pero con nombres	Alto 100%	La metodología propone el 100% de los roles necesarios para el desarrollo de las distribuciones de GNU/Linux Nova, especificando las competencias y

	diferentes. La metodología solo permite representar el 68.7% de los roles necesarios para desarrollar la distribución de GNU/Linux Nova.		habilidades exigidas para cada uno. Por tanto, se elimina la duplicación y la no existencia de roles. Ejemplo de nuevos roles son el Mantenedor de paquetes, Administrador del kernel, Comunidad de usuarios, Documentador, Administrador de servicios telemáticos y Coordinador de la metodología.

3.2.3 Indicador 3 Nivel de adecuación de las actividades de gestión de requisitos

Tabla 17: Actividades de gestión de requisitos [Fuente: Elaboración Propia]

Antes		***Después***	
Bajo 28,5%	Se pudo constatar que los procesos de ingeniería de requisitos en cada uno de los proyectos auditados se llevan a cabo de forma leve. Solo se identifican y especifican los requisitos de la distribución Nova. Se obvian procesos importantes como la administración de los cambios que estos puedan sufrir, su verificación, validación y negociación.	Alto 85,7%	La metodología propone un conjunto de actividades que permiten llevar a cabo una correcta ingeniería de requisitos y artefactos adaptados a las necesidades de documentación de las distribuciones de GNU/Linux Nova en esta disciplina. Los proyectos investigados realizan las actividades propuestas por la metodología para la gestión de requisitos, evidencia de las actividades identificación, especificación, detallamiento, verificación, validación y negociación se puede encontrar en: https://repositorio.geitel.prod.uci.cu/svn/nova. Aún existen problemas con el establecimiento de la trazabilidad de los requisitos, se realiaza el reporte de trazabilidad, pero resulta complicado representar las relaciones entre los requisitos y los componentes de software.

3.2.4 Indicador 4 Nivel de adecuación de las actividades de gestión de la configuración de software.

Tabla 18: Actividades de gestión de la configuración del software [Fuente: Elaboración Propia]

Antes		***Después***	
Bajo 25%	No se conocía ni siquiera este proceso por su nombre, sino que de manera inconsciente se llevaba a cabo solo mediante el control de versiones de los productos software, obviando actividades importantes como la definición de los elementos de la configuración y la manera de gestionar los cambios.	Alto 75%	La metodología propone un conjunto de actividades que permiten llevar a cabo una correcta gestión de la configuración y artefactos adaptados a las necesidades de documentación de la distribución GNU/Linux Nova en esta disciplina. Ejecutando las actividades propuestas existen hoy los procedimientos para hacer los cambios a diferentes niveles, y quedaron definidos los nomencladores para los elementos

			de la configuración de software, así como se estandarizaron en todos los proyectos la utilización de las mismas herramientas de control de versiones SUBVERSION, REPREPO y APACHE. No se realiza de manera directa auditorías a la configuración de software. La evidencia de las actividades ejecutadas se puede encontrar en: https://repositorio.geitel.prod.uci.cu/svn/nova y en http://nova.f10.uci.cu/

3.2.5 Indicador 5 Nivel de adecuación de las actividades de dirección de proyectos

Tabla 19: Actividades de Dirección de Proyectos [Fuente: Elaboración Propia]

Antes		***Después***	
Bajo 50%	La dirección de proyecto se lleva a cabo de manera ineficiente, pues no se tiene un nivel de conocimiento adecuado del tema. Solo se realizan parcialmente actividades relacionadas con el inicio, planificación y cierre del proyecto.	Alto 75%	La metodología propone un conjunto de actividades que permiten llevar a cabo una correcta gestión de proyectos y artefactos adaptados a las necesidades de documentación de la distribución GNU/Linux Nova en esta disciplina. Por tanto, los proyectos involucrados tienen definidos de manera correcta sus planes de desarrollo de software de acuerdo a sus requisitos y su realidad, apoyando esto, la correcta asignación de tareas y recursos para el cumplimiento de los objetivos planteados. Hoy, se trabaja en función de las planificaciones hechas. En esta actividad no solo influye la metodología, sino también la utilización de la herramienta GESPRO que permite agilizar y evaluar respecto a una serie de indicadores que ofrecen elementos para la toma de desiciones. Evidencias sobre la ejecución de estas actividades se puede encontrar en: https://repositorio.geitel.prod.uci.cu/svn/nova y en los reportes de estado de proyecto que se obtienen de la herramienta GESPRO. Aún existen dificultades con las actividades formación del personal por cuestiones relacionadas con la disponibilidad de tiempo y con el establecimiento de acuerdos con proveedores de piezas de software que se obtienen de las comunidades de software libre.

3.2.6 Indicador 6 Nivel de adecuación de las actividades de gestión de la calidad de software.

Tabla 20: Actividades de gestión de la calidad del software [Fuente: Elaboración Propia]

Antes		***Después***	
Bajo 25%	No se tienen en cuenta las actividades de aseguramiento de la calidad, solo se implementan a bajo nivel pruebas a las distribuciones GNU/Linux Nova, siendo estas poco priorizadas y por tanto afectadas por los cronogramas inexactos.	Alto 100%	La metodología propone un conjunto de actividades que permiten llevar a cabo una correcta gestión de la calidad de software y artefactos adaptados a las necesidades de documentación de las distribuciones GNU/Linux en esta disciplina. Los proyectos involucrados tienen definido el plan de aseguramiento de la calidad, que atendiendo al principio de trabajo colaborativo ha sido discutido, perfeccionado y asumido por todos. Estos proyectos realizan su cronograma teniendo en cuenta la ejecución de actividades de gestión de calidad enfocadas al proceso y al producto. La metodología propone una disciplina enfocada a las pruebas lo que ha ayudado a tener baterías completas de pruebas para probar el factor funcionalidad en función de los requisitos definidos, y el rendimiento a través de la herramienta PHORONIX TEST SUITE. Dichas pruebas han sido generalizadas a la dirección de Calidad para la liberación de estos productos. La actividad *Poner distribución GNU/Linux a disposición de la comunidad de usuarios* propició entregar a la comunidad de usuarios diferentes versiones de Nova, estimulando la detección de errores mediante la realización de concursos. Además se realizan revisiones a los procesos de desarrollo de los proyectos de Nova quedando registradas las no conformidades detectadas en sus respectivos Registros de Evaluaciones, siendo estas arregladas, seguidas y escaladas de ser necesario.

3.2.7 Indicador 7 Grado de utilización del expediente de proyecto.

Tabla 21: Utilización del expediente de proyecto [Fuente: Elaboración Propia]

Antes		***Después***	
Bajo 50%	En los expedientes de proyectos auditados se detectaron un total de 69	Alto 100%	La metodología propone un expediente de proyecto basado en el nivel 2 de

	no conformidades todas críticas pues se evidenció que no existe una cultura de documentar a la par que se desarrolla.		CMMI que abarca todo el ciclo de vida de una distribución GNU/Linux, lo que facilita su utilización. El mismo ha sido completado por los proyectos involucrados hasta la actual fase de construcción en la que se encuentran, logrando mayor comunicación entre los desarrolladores y garantizando guardar las tendencias y lecciones aprendidas en el proyecto. Queda trabajar más en fomentar una cultura organizacional donde los desarrolladores vean la importancia de documentar sus procesos en el expediente de proyecto. Los expedientes de los proyectos Nova Base, Nova Escritorio, Nova Servidores y Nova Ligero se encuentran en : https://repositorio.geitel.prod.uci.cu/svn/nova

La metodología ha sido implementada de manera parcial porque no ha transcurrido el tiempo necesario para cubrir el ciclo de vida de la distribución GNU/Linux Nova, pero los resultados obtenidos muestran que es capaz de disminuir las desviaciones detectadas siempre que los involucrados estén comprometidos con las prácticas y principios que propone.

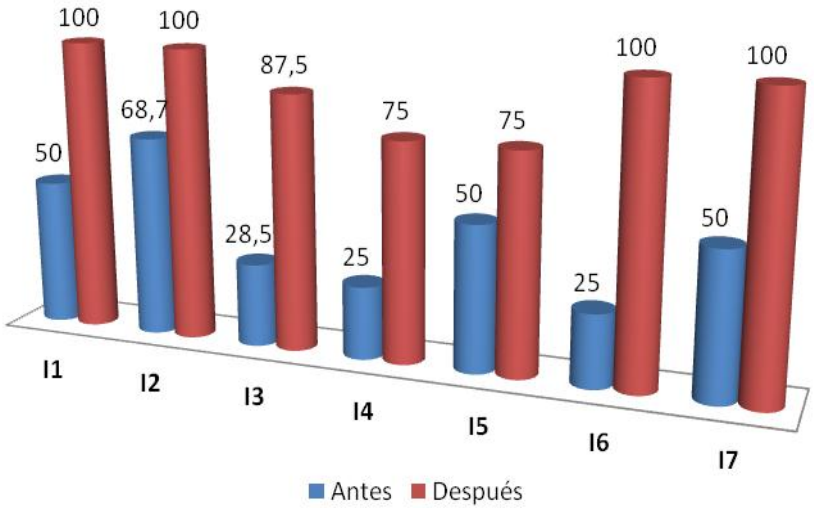


Figura 18 Resultado de evaluar la metodología antes y después de existir

La figura 18 representa un resumen de la situación antes y después de existir la metodología. Las notas encima de cada columna representan los valores que han tomado los indicadores evaluados. La metodología ha sido aceptada positivamente por los desarrolladores. Además las desviaciones

detectadas inicialmente han sido eliminadas en un 90% mostrando mejoría el proceso de desarrollo de la distribución GNU/Linux Nova. Teniendo en cuenta los resultados se rechaza la hipótesis nula y se prueba la veracidad de la hipótesis de la investigación.

3.3 Evaluación mediante el método SCAMPI C

Figura 19 Nivel de implementación de las áreas de procesos del nivel 2 de CMMI [32]

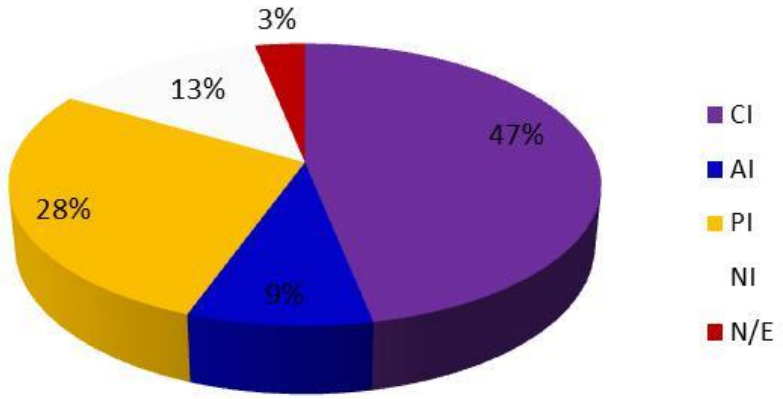


Figura 20 Distribución de los niveles de implementación de las prácticas del nivel 2 de CMMI [32]

Tomando como muestra el proyecto Nova Escritorio la dirección de Calidad de la Universidad aplicó al mismo una evaluación mediante el método SCAMPI C que permite medir el grado de implementación de las áreas de procesos de CMMI para alcanzar el nivel 2. Se obtienen los resultados que se aprecian en las figuras 19 y 20. Estos demuestran que la metodología sirve de apoyo para alcanzar el nivel 2 de CMMI con el correcto nivel de implementación de la misma. Cada área de proceso ha sido implementada de manera mediana y alta (Ver Anexo 5). Sumando a esto que se ha implementado más de la mitad de las prácticas y un 28% de estas de forma parcial. (Ver Anexo 5)

3.4 Comparación de la metodología Nova – Open con otras propuestas similares

En el mundo del desarrollo de software existen un gran número de guías de buenas prácticas para

obtener productos de calidad, en tiempo y bajo los costos establecidos, sin embargo, el éxito está en su correcta elección e implementación según las características de los pilares fundamentales de un proceso de desarrollo de software.

La siguiente tabla muestra una comparación de exitosas y reconocidas guías de desarrollo de software a nivel mundial con la metodología Nova – OpenUp, siguiendo la finalidad de identificar fortalezas y debilidades de esta respecto al desarrollo de la distribución cubana de GNU/Linux Nova.

Se selecciona para la comparación PMBOK por ser una de las guías más reconocidas para las actividades de gestión de proyectos, Scrum porque según reportes de la organización Version One es la metodología más utilizada a nivel mundial, CMMI por ser un prestigioso y exigente modelo de calidad de software que brinda grandes potencialidades, OpenUp por ser la metodología más adecuada de las estudiadas en el capítulo 1 para ser transformada en función del desarrollo de la distribución de GNU/Linux Nova, SXP porque se utilizaba anteriormente en este desarrollo y las LFS por ser una de las guías más utilizadas para el desarrollo de distribuciones de GNU/Linux.

Los indicadores a medir responden a las desviaciones identificadas que dieron origen a esta investigación. Para medir los indicadores se utiliza la leyenda Alta (Contiene todas las actividades necesarias para desarrollar el indicador), Media (Contiene parcialmente las actividades necesarias para desarrollar el indicador) y Baja (No contiene las actividades necesarias para desarrollar el indicador).

Tabla 22: Comparación de Nova - OpenUp con otras propuestas [Fuente: Elaboración Propia]

Indicadores	**PMBOK**	**Scrum**	**CMMI**	**LFS**	**Nova - OpenUp**	**OpenUp**	**SXP**
Preparación del personal	Alta	Baja	Media	Baja	Alta	Media	Baja
Gestión de requisitos	Media	Alta	Alta	Baja	Alta	Media	Alta
Gestión de la configuración de software	Baja	Alta	Alta	Baja	Alta	Media	Media
Gestión de la calidad	Alta	Media	Alta	Baja	Alta	Media	Baja
Documentación	Alta	Media	Alta	Baja	Alta	Media	Baja
Desarrollo de distribuciones GNU/Linux	Baja	Baja	Baja	Alta	Media	Baja	Baja
Gestión de proyectos	Alta	Baja	Media	Baja	Media	Media	Media

Se puede observar que la metodología Nova – OpenUp cubre todos los indicadores medidos para el desarrollo de la distribución Nova, sin embargo, se recomienda sea complementada su utilización con otras guías más completas en áreas como la gestión de proyectos y el desarrollo técnico del

producto.

3.5 Análisis económico de la implementación de la metodología Nova – OpenUp

Antes de poner en práctica la metodología se hizo un análisis de las probabilidades de que mejorara el proceso de desarrollo de Nova. Posteriormente se determinó que los equipos de desarrollo de las distribuciones de GNU/Linux Nova estaban listos para afrontar la migración de metodología de desarrollo de software. Dicha metodología debe favorecer los procesos que se ejecutan porque:

- Posee actividades en función de la preparación de los roles propuestos, lo que va a minimizar a mediano plazo la sobrecarga de tareas en una sola persona y por tanto debe aumentar la velocidad de desarrollo de software.
- Incluye a las comunidades de usuarios como parte del desarrollo del software, lo que minimiza los costos teniendo en cuenta que las mismas están formadas por personas que trabajan de manera voluntaria sin recibir retribución alguna.
- El trabajo colaborativo que promueve debe fortalecer la reutilización de componentes de software y la socialización del conocimiento que se adquiere en el desarrollo de software.
- Aporta actividades de dirección de proyecto que deben minimizar la ocurrencia de riesgos, estabilizando las planificaciones.
- Aumenta la calidad del software mediante la ejecución de pruebas y actividades sistemáticas de aseguramiento de la calidad.

Durante la implementación de la metodología se realizó una planificación que contaba con las actividades que se muestran en la tabla 16.

Tabla 23: Costos directos de implementar la metodología [Fuente: Elaboración Propia]

Actividades	Duración	Recursos necesarios	Costo Estimado
Auditoría al proyecto Nova Escritorio.	20 horas	1 profesor 5 estudiantes	$90.00
Auditoría al proyecto Nova Ligero.	20 horas	1 profesor 5 estudiantes	$90.00
Auditoría al proyecto Nova Servidores.	20 horas	1 profesor 5 estudiantes	$90.00
Auditoría al proyecto Nova Base.	20 horas	1 profesor 5 estudiantes	$90.00
Análisis de los resultados	8 horas	1 profesor	$36.00
Presentación de los resultados	2 horas	12 especialistas 5 profesores	$108.00
Diseño de la metodología	2304	1 profesor	$10368.00

	horas		
Análisis de la metodología con los desarrolladores involucrados	4 horas	12 especialistas 5 profesores	$ 216.00
Capacitación sobre cómo utilizar la metodología	18 horas	12 especialistas 5 profesores	$ 972.00
Implementación parcial de la metodología	576 horas	6 profesores 5 estudiantes	$15552.00
Auditoría al proyecto Nova Escritorio.	20 horas	1 profesor 5 estudiantes	$90.00
Auditoría al proyecto Nova Ligero.	20 horas	1 profesor 5 estudiantes	$90.00
Auditoría al proyecto Nova Servidores.	20 horas	1 profesor 5 estudiantes	$90.00
Auditoría al proyecto Nova Base.	20 horas	1 profesor 5 estudiantes	$90.00
Costo total de mano de obra	$27972.00		

Tabla 24: Costo de materiales utilizados [Fuente: Elaboración Propia]

Tipo de materiales	**U/M**	**Cantidad**	**Precio**
Papel	Paquete	1	4.44 CUC
Plumón de pizarra	Unidad	1	1.02 CUC
Bolígrafo	Unidad	15	7.50 CUC
Computadoras	Unidad	15	4 500 CUC
Costo directo total de materiales utilizados		$112824.00	

Tabla 25: Resumen de costos [Fuente: Elaboración Propia]

Costo total de mano de obra	$27972.00
Costo directo total de materiales utilizados	$112824.00
Resumen de costos	$140796.00

La implementación de la metodología no trajo consigo costos indirectos asociados. Para el departamento Sistemas Operativos el costo de asumir la metodología parcialmente fue de 140796 pesos, se debe tener en cuenta que junto a las actividades de implantación de la metodología también se desarrollaba la distribución de GNU/Linux Nova. La metodología tiene gran impacto social porque estimula el desarrollo de software teniendo en cuenta principios como la soberanía tecnológica, seguridad, socio-adaptabilidad y sostenibilidad durante el proceso de desarrollo de la

distribución cubana GNU/Linux Nova. Promueve el trabajo colaborativo entre las comunidades de usuarios y los desarrolladores, la capacidad de tomar decisiones sobre los procesos a realizar, la flexibilidad y adaptabilidad al entorno de desarrollo de Nova. Lo antes expresado se ve reflejado en los lineamientos 131, 132 y 135 emitidos en la política económica y social de Cuba. Estos se refieren a la política de ciencia, tecnología e innovación y apoyan la política de migración del país a software libre y la innovación tecnológica a través de la utilización de estándades de calidad apropiados para la producción de software.

Conclusiones del Capítulo 3

- Según el Modelo de evaluación de metodologías propuesto por la Universidad Andrés Bello, Nova – OpenUp es correcta estructuralmente porque cumple con las variables que debe tener en cuenta una metodología de desarrollo de software.
- Según los especialistas encuestados la metodología es capaz de afrontar el desarrollo de la distribución de GNU/Linux Nova, el 100% de estos opina que la misma es importante para el desarrollo de este producto.
- La metodología se implementó parcialmente en los proyectos Nova Escritorio, Nova Ligero, Nova Servidores y Nova Base, evidenciando que ha apoyado la disminución de las desviaciones detectadas en un 90%.
- Se comprobó mediante resultados de evaluación según el método SCAMPI C que la metodología Nova – OpenUp favorece la implementación de las áreas de procesos para alcanzar el nivel 2 de CMMI.
- La metodología necesita ser complementada con prácticas de gestión de proyecto y otras técnicas para el desarrollo de distribuciones de GNU/Linux.

CONCLUSIONES

En la presente investigación se llega a las siguientes conclusiones:

- Los procesos de desarrollo de las principales distribuciones de GNU/Linux, no constituyen una referencia completa para establecer el proceso de desarrollo de la distribución de GNU/Linux Nova.
- No existe metodología, modelo o norma capaz de integrar todos los procesos necesarios para el desarrollo de distribuciones de GNU/Linux.
- La implementación parcial de la metodología OpenUp con cambios particulares en sus disciplinas, en correspondencia con las características de la distribución GNU/Linux Nova, permitió la eliminación de las desviaciones identificadas en el 2010 en un 90%.
- La metodología Nova – OpenUp es correcta e importante para el desarrollo de las distribuciones de GNU/Linux Nova.
- La metodología Nova - OpenUp para el desarrollo de la distribución GNU/Linux Nova apoya positivamente la obtención del nivel 2 de CMMI.

RECOMENDACIONES

- Perfeccionar la metodología Nova - OpenUp a través de las lecciones que se aprenden durante su implementación para lograr que sea más completa y útil para usuarios noveles.
- Proveer actividades para las disciplinas de despliegue y soporte de las distribuciones GNU/Linux.
- Proveer las actividades que guíen hacia la gestión de los servicios de la distribución de GNU/Linux Nova.

BIBLIOGRAFÍA

1. SERGIO. *¿Qué son las distribuciones de GNU/Linux?*.[En línea] junio 2009 Disponible en: http://www.pctux.com.ar/2009/06/%C2%BFque-son-las-distribuciones-de-gnulinux.html. Consultado Septiembre 2011.
2. FIRVIDAS, Abel; HERRERA, Dariem; ALBALAT, Miguel; MACHIN, Jorge L; HERRERA, Anielkis; FERNANDEZ, Yusleydi; CASTRO, Monica; MIRANDA, Raydel; SOLER, Yuniel. *Nova 3.0 avances y expectativas de la distribución cubana de GNU/Linux*. [En línea] Cuba. Febrero de 2011. Disponible en: www.informaticahabana.cu/node/1173. Consultado 8 de abril de 2011
3. FERNANDEZ DEL MONTE, Yusleydi; CASTRO ALBO, Mónica. *Resultados de la segunda auditoría interna hecha al proyecto Nova*. [Soporte digital] Cuba. Enero de 2011. Consultado Abril de 2011.
4. Proyecto Teruel Digital. ¿Qué es una distribución de GNU/Linux?. [En línea] 2006. Disponible en: http://tirwal.terueldigital.es/doc-manual-usuario-tirwal/ch01s04.html. Consultado Junio de 2012.
5. Wikipedia. *Distribución Linux*. [En línea] 2012. Disponible en: http://es.wikipedia.org/wiki/Distribuci%C3%B3n_Linux. Consultado Junio de 2012.
6. Ecured. *Distribución de GNU/Linux*. [En línea] Cuba. 2012. Disponible en: http://www.ecured.cu/index.php/Distribuci%C3%B3n_de_GNU/Linux. Consultado Junio de 2012.
7. PIERRA FUENTES, Allan. *Conceptualización y Reestructuración estratégica de la Distribución Cubana GNU/Linux Nova*. [Tesis de Maestría] Universidad de las Ciencias Informáticas. Cuba. 2011.
8. Blog de Ubuntu. *¿Qué es una distribución GNU Linux?* [En línea] Disponible en: http://120linux.com/distribuciones/. Consultado Junio de 2012.
9. Comités miembros de ISO. *Sistemas de gestión de la calidad — Requisitos*. [Soporte digital]. 2000. Consultado en marzo 2011.
10. BEKMANS, Gerard. *Linux From Scratch Version 7.0*. [En línea] 1999. Disponible en: http://www.linuxfromscratch.org/lfs/view/7.0/. Consultado en marzo 2012.
11. Fedora. *Sitio oficial de Fedora*. [En línea] 2012. Disponible en: http://fedoraproject.org/es/. Consultado en mayo 2012.
12. Debian. *Sitio oficial de Debian*. [En línea] 2012. Disponible en: http://www.debian.org/index.es.html. Consultado en mayo 2012.
13. Proyecto Nova. *Concepto Nova. Reestructuración Estratégica*. [Soporte digital]. Cuba. 2010. Consultado en marzo 2011.
14. PEÑALVER, Gladys. MENESES, Abel. GARCÍA, Sergio. SXP*, metodología ágil para el desarrollo de software.* [En línea] Cuba. 2010. Disponible en:

http://usbvirtual.usbcali.edu.co/ijpm/images/stories/documentos/.../009.pdf. *Consultado en noviembre de 2012.*

15. Equipo Del Proyecto De Actualización 2004 De La Guía Del PMBOK. *Guía de los Fundamentos de la Dirección de Proyecto*. [Soporte digital] Tercera Edición. Editorial Project Management Institute, Inc. Four Campus Boulevard. Newtown Square, Pennsylvania. EEUU. 2004. 409 págs. ISBN: 1-930699-73-5 Consultado en abril del 2011.
16. GABARDINI, Juan. CAMPOS, Lucas. *Balanceo de Metodologías Orientadas al Plan y Ágiles. Herramientas para la Selección y Adaptación.* [En línea] Disponible en: http://evapostgrado.uci.cu/mod/resource/view.php?id=10660. Consultado en marzo 2011.
17. TINOCO GÓMEZ, Oscar. ROSALES LÓPEZ, Pedro P. SALAS BACALLA, Julio. Revista de la Facultad de Ingeniería Industrial 13(1): 70-74. *Criterios de selección de metodologías de desarrollo de software*. [Soporte digital]. 2010. ISSN: 1810-9993. Consultado en marzo 2011.
18. KRUCHTEN, Philippe. *The Rational Unified Process An Introduction.* [Soporte digital] Tercera Edición. Editorial. Addison Wesley. 2003. 336 pág. ISBN 0-321-19770-4. Consultado en abril del 2011.
19. Rational Unified Process. *Ayuda de RUP*. [En línea] 2006. Disponible en: http://eva.uci.cu. Consultado septiembre 2011.
20. RAMIREZ NOEL, Yenia. ACOSTA GISPERT, Alianna. *Propuesta de Mejora de CMMI Nivel 2 de Madurez para Proyectos de Desarrollo Ágil con metodología eXtreme Programming en la Universidad de Ciencias Informáticas*. [Tesis de grado] Universidad de las Ciencias Informáticas. Cuba. 2010.
21. PALACIO, Juan. *Flexibilidad con SCRUM.* Edición de Octubre 2007. 2007. 192 págs. [En línea] Disponible en: http://eva.uci.cu. Consultado en abril del 2011.
22. MOUSQUES, Gastón. *Metodología FDD*. [En línea] 2003. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=9303&subdir=/Metodologias/FDD Consultado Junio de 2011.
23. *CHICAIZA AYALA, Alexandra P. Desarrollo de software de nomina de empleados utilizando la metodologia crystal. [En línea] 2007. Disponible en: http://*repositorio.espe.edu.ec/bitstream/21000/556/1/T-ESPE-021804.pdf *Consultado Mayo de 2012.*
24. PEÑALVER, Gladys. MENESES, Abel. GARCÍA, Sergio. SXP*, metodología ágil para el desarrollo de software.* [En línea] Cuba. 2010. Disponible en: http://usbvirtual.usbcali.edu.co/ijpm/images/stories/documentos/.../009.pdf. *Consultado en noviembre de 2012.*

25. IEEE Computer Society Professional Practices Committee. *Guide to the Software Engineering Body of Knowledge*. [En línea] 2004. Disponible en: http://eva.uci.cu. Consultado septiembre 2011
26. CaliSoft. *Programa de Mejora*. Cuba. 2011. [En línea] Disponible en: http://calisoft.uci.cu/index.php/proceso-de-mejora/46. Consultado septiembre 2011.
27. GESPRO 13.05. *¿Qué es Gespro?* Cuba. 2012 [En línea] Disponible en: http://gespro.prod.uci.cu/ Consultado Junio 2013
28. Zentyal. *TIC'S para PYMES*. 2012. [En línea] Disponible en: http://www.zentyal.com/es/small-business-it/. Consultado Junio 2013
29. Geany. 2008. [En línea] Disponible en:http://www.geany.org/ Consultado Junio 2013
30. CaliSoft. Programa de Mejora. *0516_ROLES Y RESPONSABILIDADES* [En línea] Cuba. 2009. Disponible en: http://calisoft.uci.cu. *Consultado Mayo de 2012.*
31. FERNANDEZ, Yusleydi. GUERRERO, Sonia. FIRVIDAS, Abel. HERRERA, Dariem; ALBALAT, Miguel. MACHIN, Jorge L. CASTRO, Monica. HURTADO, Mijail. *Roles y competencias necesarios para los desarrollos del proyecto Nova.* [Soporte digital] Cuba. Abril 2011. Consultado Mayo de 2012.
32. CALISOFT. *Resultados del SCAMPI.* [Soporte digital] Cuba. Enero 2013. Consultado Enero de 2013.
33. Implantación de Aplicaciones Informáticas de Gestión. *¿Qué son los repositorios?* [En línea] 15 de septiembre de 2011. Disponible en: http://informatica.gonzalonazareno.org/plataforma/mod/page/view.php?id=930. Consultado noviembre 2011.
34. *¿Qué es un sistema operativo en modo texto?* [En línea] Disponible en: http://lrggrupo32.blogspot.com/2009/12/que-es-un-sistema-operativo-en-modo.html. Consultado septiembre 2011.
35. FERNANDEZ DEL MONTE, Yusleydi. GUERRERO LAMBERT, Sonia. *Capacitación orientada a eliminar deficiencias en el aprendizaje. Sistema de Gestión de Conocimientos*. [Tesis de grado] Universidad de las Ciencias Informáticas. Cuba. 2008.
36. PRESSMAN, Roger. *Software Engineering. A practitioner's approach*. [Soporte digital] Septima Edición. Editorial McGraw-Hill. New York. 2010. 889 págs. ISBN 978-0-07-337597. Consultado en abril del 2011.
37. MARTÍNEZ, Alejandro. MARTÍNEZ Raul. *Guía a Rational Unified Process.* [Soporte digital] Consultado en marzo 2011.
38. The Eclipse Foundation. *OpenUp*. [En línea] 2012. Disponible en: http://epf.eclipse.org/wikis/openup/index.htm. Consultado Mayo de 2012.

39. Especialistas de Calisoft. *Proceso de aseguramiento de la calidad. Diplomado de Auditorías y Revisiones de software*. [Soporte digital]. Cuba. 2009. Consultado en marzo 2011.
40. Comité coordinador de estándares de la organización IEEE. *Standard Glossary of Software Engineering Terminology*. [Soporte digital]. 1990. ISBN 0-7381-0391-8, SS13748. Consultado en marzo 2011.
41. MEGÍAS, David; AYCART, David; GIBERT, Marc; HERNÁNDEZ, Jordi. *Ingeniería de software en entornos de SL*. UOC. [Soporte digital] Segunda Edición. Editorial Eureca Media. Universidad Oberta de Catalunya. 2007. 314 pág. serie XP06/M2112/01486. Consultado en abril del 2011.
42. HERRERA, Dariem. GUERRERO, Sonia. FERNÁNDEZ, Yusleydi. ALBALAT, Miguel. MACHÍN, Jorge L. PÉREZ, Hector. QUEVEDO, Ricardo. Revista Cubana de Ciencias Informáticas. *Estado actual de los sistemas de construcción de paquetes en diferentes distribuciones de GNU/Linux*. [En línea] Cuba. 2012. Disponible en: http://publicaciones.uci.cu/index.php/rcci/index. Vol. 6 No. 2. Consultado Noviembre de 2012.
43. ALBO CASTRO, Monica M. Proceso de pruebas de la distribución cubana GNU/Linux Nova. [Soporte digital] Cuba. 2011. Consultado Mayo de 2012.
44. Ubuntu. *Repositorios*. [En línea] Agosto 2011. Disponible en: https://help.ubuntu.com/community/Repositories/Ubuntu. Consultado septiembre 2011.
45. *Compilación de software con Debian Linux*. [En línea]. Disponible en: http://www.aboutdebian.com/compile.htm. Creado en el 2003. Consultado septiembre 2011.
46. CaliSoft. Programa de Mejora. *Expediente de proyecto del PM 3.3 Parte 1*. 2011. [En línea] Disponible en: http://calisoft.uci.cu/index.php/proceso-de-mejora/46. Consultado septiembre 2011.
47. CaliSoft. Programa de Mejora. *Expediente de proyecto del PM 3.3 Parte 2*. Cuba. 2011. [En línea] Disponible en: http://calisoft.uci.cu/index.php/proceso-de-mejora/46. Consultado septiembre 2011.
48. AMOR IGLESIAS, Juan José. HERRAIZ TABERNERO, Israel. ROBLES MARTÍNEZ, Gregorio. *Desarrollo de proyectos de Software Libre*. [En línea] 2007 Disponible en: www.uoc.edu. Consultado en Septiembre 2011.
49. RODIN, Josip. *Debian New Maintainers' Guide*. [En línea] 2010. Disponible en: http://doc.ubuntu-es.org/Gu%C3%ADa_de_empaquetamiento/Completa. Consultado noviembre 2011.
50. *Mejores distribuciones gnu/linux***.** [En línea] Disponible en: http://www.taringa.net/posts/linux/1214674/Las-10-mejores-distribuciones-de-linux_.html. Consultado noviembre 2011.
51. GERONIMO. Calidad en el desarrollo de software. [En línea] 7 de Marzo de 2008. Disponible en: http://www.geronet.com.ar/?p=23. Consultado: 8 de abril de 2011.
52. GARCIA GARCIA, Guillermo. *Historia de Linux y sus distribuciones*. [En línea] 2008. Disponible en:

http://www.cdlibre.org/clase/0506amaya/0506_7l/guillermo_garcia/enlaces1/linux.html#L149. Consultado 10 de abril de 2011.

53. FERNANDEZ SANZ, Luis. *Necesidades de la medición en la gestión y el aseguramiento de la calidad del software*. [En línea] 14 de Enero de 1999. Disponible en: http://www.sc.ehu.es/jiwdocoj/remis/docs/aseguracal.htm. Consultado Marzo de 2011.
54. Diccionario de definiciones. *Definición de modelo de gestión*. [En línea] 2008. Disponible en: http://definicion.de/modelo-de-gestion/. Consultado 17 de Abril de 2011.
55. Comité coordinador de estándares de la organización IEEE. Standard for Information Technology—Software life cycle processes. [Soporte digital] 1990 ISBN 0-7381-0428-0, SS94581. Consultado en marzo 2011
56. Steering Group, Product Team y Configuration Control Board. *CMMI, Guía para la integración de procesos y la mejora de producto*s. [Soporte digital] Segunda edición. Editorial Cátedra de Mejora de Procesos de Software en el Espacio. Universidad Politécnica de Madrid. 2009. ISBN 9788478290963. Consultado en marzo 2011
57. Ubuntu. *Sitio oficial de Ubuntu*. [En línea] 2012. Disponible en: http://www.ubuntu-es.org/. Consultado en mayo 2012.
58. MÉNDEZ NAVA, Elvia Margarita. Modelo de evaluación de metodologías para el desarrollo de software. [Tesis de especialidad] Universidad Católica Andrés Bello. Venezuela. 2006.
59. SARASA, Manel. *Software libre 2.0*. [En línea] Consultado septiembre 2011.

ANEXOS

Anexo 1: Guía de entrevista para determinar las causas de las desviaciones detectadas en los proyectos Nova Escritorio, Nova Servidores, Nova Base y Nova Ligero.

1. Exponga sus criterios sobre las causas de las desviaciones detectadas en la auditoría.
2. ¿Qué elementos cree que hayan influido en las causas que dan origen a las desviaciones existentes?

Anexo 2: Guía de observación

La guía de observación que se tiene en cuenta está compuesta de los siguientes indicadores:

1. Procesos de ingeniería de requisitos
2. Procesos de generación de repositorios
3. Procesos de pruebas
4. Procesos de gestión de proyectos
5. Procesos de gestión de ambiente

Anexo 3: Encuesta para la evaluación de la metodología Nova - OpenUp

1. A partir de las variables que se muestran en la siguiente tabla:

a) Otorgue a cada una teniendo en cuenta la metodología Nova - OpenUp y el proceso de desarrollo de distribuciones GNU/Linux una nota entre 5 y 1, sabiendo que 5 es el mayor grado en que la metodología cumple la variable y 1 el menor.

b) Para cada valor que otorgue igual o inferior a 3 Justifique su respuesta.

Tabla 26: Variables a evaluar [Elaboración Propia]

Variables	Nota Otorgada	Justificación
Ajuste de la metodología a los objetivos		
Existencia de actividades destinadas a definir el sistema		
Apoyo a la comunicación efectiva entre los involucrados.		
Existencia de un entorno dinámico orientado al usuario		
Especificación los responsables de los resultados		
Puede ser aprendida		
Soporte de herramientas Case		
Soporte de la eventual evolución del sistema		
Enfoque a la calidad de software		
Actividades para el trabajo en entornos de software libre		
Integración de procesos para el desarrollo de distribuciones de GNU/Linux		

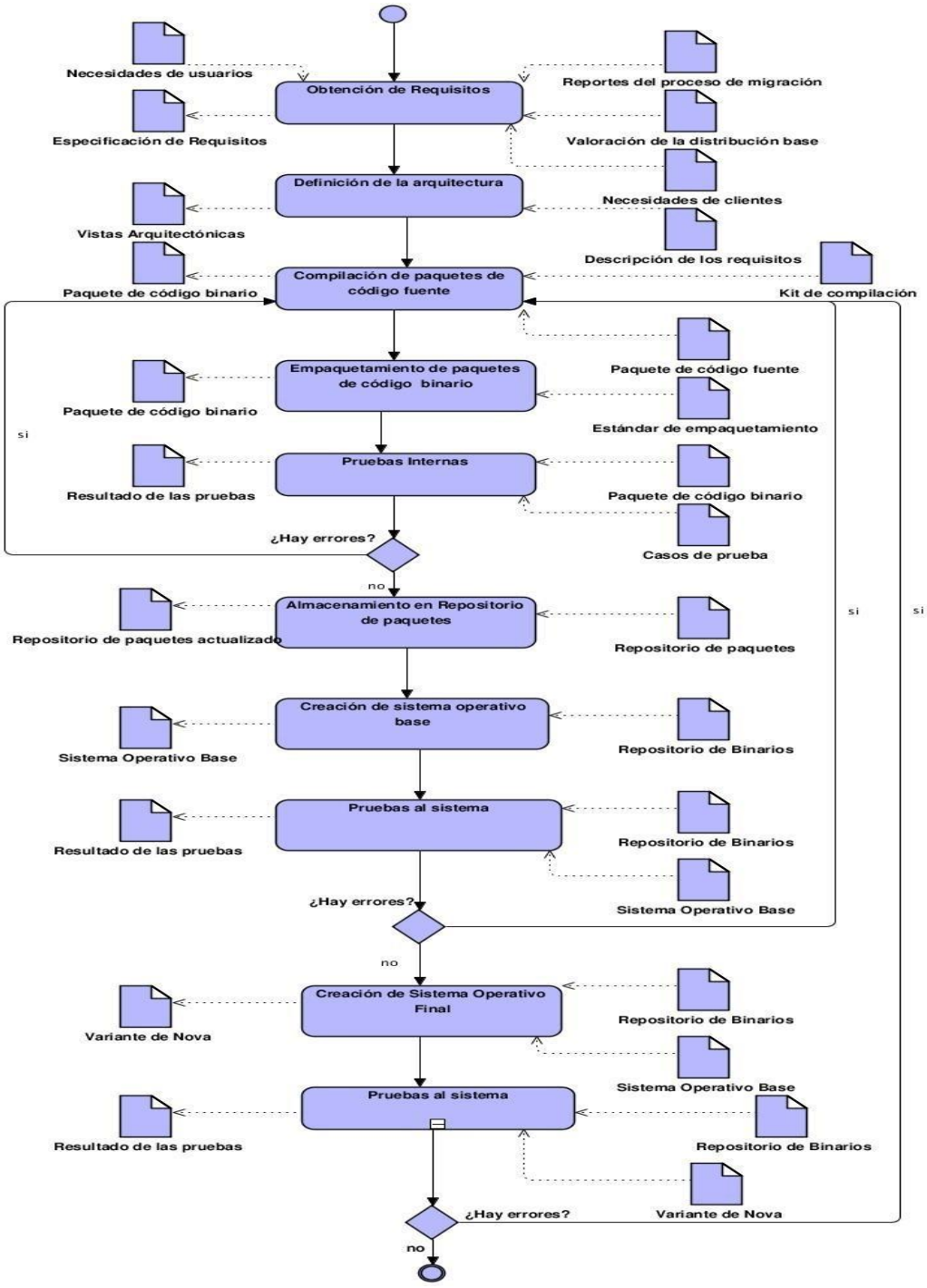
2. ¿Cree importante la utilización de la metodología en el desarrollo de distribuciones GNU/Linux? Si_____ No_____ ¿Por qué?

3. ¿Cree que la metodología es completa? Si_____ No_____ ¿Por qué?
4. ¿Cree que la metodología es suficiente para afrontar el desarrollo de distribuciones GNU/Linux? Si_____ No_____ ¿Por qué?

5. ¿Es correcta la metodología? Si_____ No_____ ¿Por qué?

6. ¿Considera compleja la implementación de la metodología en un entorno real de desarrollo? Si_____ No_____ ¿Por qué?

Anexo 4: Proceso de desarrollo de la distribución GNU/Linux Nova



Anexo 5: Guía para la interpretación para el método SCAMPI

	Nivel de implementación alto	75-100 %
	Nivel de implementación medio	25-74 %
	Nivel de implementación bajo	0-24 %

Color	Código	Descripción
	CI	Completamente Implementado
	AI	Ampliamente Implementado
	PI	Parcialmente Implementado
	NI	No Implementado
	N/E	No Evaluado

Valores: Porciento de prácticas por niveles de implementación.

Anexo 6: Modelo de desarrollo de la distribución de GNU/Linux Nova.