Universidad de las Ciencias Informáticas

Facultad 1

Título: *Procesos de ingeniería de requisitos en el desarrollo de distribuciones de GNU/Linux*

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autoras: *Saily Llechú Ramos / Dainet Torres Torres*

Tutora: *Ing. Yusleydi Fernández del Monte*

DECLARACIÓN DE AUTORÍA

Declaramos ser autoras de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____ _____

Firma del Autor Firma del Autor

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor: Ing. Yusleydi Fernández del Monte.

- Graduada en el año 2008 de Ingeniero en Ciencias Informáticas.
- Analista de software del Dpto.SODTL
- Administradora principal de la calidad de software del Dpto.SODTL
- Profesora del Dpto. Ingeniería y Gestión de Software
- Tiene 4 años de experiencia en el trabajo vinculado con la Calidad de Software.
- Tiene 3 años de experiencia en el trabajo vinculado con la Gestión de Conocimientos.
- Tiene 3 años de experiencia en la docencia universitaria.
- Ha publicado y realizado presentaciones en eventos nacionales e internacionales sobre temas de gestión de conocimientos y calidad de software.

Agradecimientos

Saily

A mi mamá por ser mi madre y la mejor de todas, por su entera confianza, dedicación, entrega, por ser mi mejor amiga, mi consejera, por estar siempre cuando la necesito, por darme aliento cuando sentía que se acababa el aire, por apoyarme durante toda mi vida, por guiar cada uno de mis pasos, por encaminar mi futuro hoy y siempre ya que en cada uno de mis logros ella es y será la fuerza para alcanzarlos y por encima de todo por amarme con ese amor que solo ella es capaz de darme. A mi papá por ser el mejor de todos, por apoyar y respetar mis decisiones aun cuando fueron erróneas. Por aconsejarme y estar ahí para mí cuando lo necesitaba. Por cuidarme y luchar por darme lo mejor sin pensar que lo mejor para mí es él y por enseñarme día a día el camino a seguir. Por los abrazos tan cálidos y fuertes que siempre recibí de los dos, donde me sentía segura, tranquila, plena. A los dos por darme la vida y hacerme mujer. A mi esposo por ser mi alma gemela, le agradezco porque con él conocí el amor. Por su apoyo, paciencia, dedicación, por llenarme con detalles porque estuvo y estará siempre cuidándome aconsejándome, por entregarme su corazón y por amar y llenar al mío minuto a minuto de eterna felicidad. A mis abuelas Coralia y mima, a mima por cuidarme desde donde está Dios y a mama por darme todo su amor y su apoyo. Al resto de mi familia por preocuparse siempre por mis estudios y por quererme de la forma tan linda como quieren. A todas mis amistades en especial a Dayane por ser mi consejera, mi hermana, por escucharme y apoyarme siempre. A mi compañera de tesis por tener mucha paciencia conmigo y por escucharme siempre y aconsejarme siempre cuando la necesité.

Agradecimientos

Dainet

A mi abuela Dignora por ser la persona por la que he luchado para lograr esta meta. A mi madre por ser por encima de cualquier cosa mi razón de existir, por estar presente en todos los momentos de mi vida, espero que esté muy orgullosa de mi como yo lo estoy de ella. A mi padre por tratar de que fuese una mejor persona, que me superase y luchara por mis sueños. A mis hermanos, Dilian, Felipe, Claudia y Maidelis a los que adoro con toda mi alma. A Ivón por ser amiga, madre, por escucharme, creer en mí, apoyarme cuando más necesitaba, por ser esa persona excepcional que siempre tendré en mi corazón sin importar el tiempo o la distancia que nos separe. A Alain, por estar conmigo y apoyarme en los peores momentos de mi vida y aún así darme fuerzas y confiar en mí. A Ercio por enseñarme a ver las partes buenas de las cosas, a Maritza. A mi segunda madre en estos momentos, mi tía Teresa la cual ha sido y es un ejemplo a seguir para mí, por su dignidad, dedicación y su manera de enfrentar la vida. A mi Tata (July) por escucharme siempre. A mi mami negra Mexy por apoyarme siempre en mis derrotas y celebrar mis victorias. A mis abuelas Leonor y Marcia. A Pacho por ser el hermano mayor que siempre quise tener. A mis amistades, Dayi, Sonia, Danne, Sergito, Alberto en especial a mi Cuki por estar conmigo en cada uno de mis días de esta universidad, apoyándome, comprendiéndome, aconsejándome y sobre todas las cosas por creer en mí cuando yo misma no lo hice, A mi compañera de tesis Sailita por tener paciencia y no perder la cabeza conmigo. Y por último aunque no menos importante a mi papá Albert por hacer de mí la persona que soy hoy, por indicarme el camino correcto y sobre todas las cosas por hacerme sentir orgullosa de ser su hija mayor.

Dedicatoria

Dedico esta tesis a mi mamita linda por ser mi ejemplo de mujer y madre a seguir. A mi papá porque siempre ha estado orgulloso de tenerme como hija. A mi tititico por creer en mí y a mis abuelas Coralia por cuidarme siempre y a mima porque aunque no me vio entrar en la Universidad se que desde donde está, está orgullosa por haber logrado este sueño.

Saily

Dedico esta tesis a mi abuela Dignora por ser mi guía en cada paso que he dado en mi vida, por inculcarme sus virtudes y ayudarme a ser cada día una mejor mujer, y sobre todas las cosas por amarme como lo hizo cuando estaba viva y por seguir mis pasos desde el cielo.

Dainet

RESUMEN

La ingeniería de software (ISW) se considera como una nueva área de la ingeniería, y la profesión de ingeniero informático es una de las más demandadas. El ciclo de vida del desarrollo del software por lo general incluye diversas fases entre las que se encuentra la fase de requisitos, por lo que surge la ingeniería de requisitos (IR) como una necesidad de contar con un enfoque metódico, sistemático y disciplinado que le permita a los desarrolladores resolver uno de los problemas principales que afecta al éxito de los proyectos, una mala gestión de requerimientos, por fallas en la adquisición, evaluación y/o documentación de los mismos así como dificultades en la comunicación y/o cambios no controlados durante el proyecto. La IR es una de las partes más importantes y menos apreciadas de la ISW. Actualmente no existen procesos de IR definidos en el desarrollo de las distribuciones GNU/Linux, esto trae como consecuencia que se capturen pobremente las especificaciones que tendrá la versión del producto a desarrollar, además de perder la trazabilidad de los requisitos entre otros factores que influyen en la administración de los mismos. Debido a la importancia que representan los mismos en el proceso de desarrollo de software (PDSW) se decide diseñar procesos de IR para las distribuciones GNU/Linux, siendo este el objetivo de la investigación. Se evalúa la propuesta mediante el método Delphy por el criterio de especialistas y por el método experimental tomando como escenario de prueba la distribución GNU/Linux Nova para determinar las ventajas, desventajas, y corroborar los beneficios de los procesos propuestos.

Palabras claves: Ingeniería, requisitos, procesos, distribución, desarrollo, escenario de prueba.

.

TABLA DE CONTENIDO

TABLA DE CONTENIDO

Introducción

"*La parte más difícil de construir un sistema es precisamente saber qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallados, incluyendo todas las interfaces con gente, máquinas y otros sistemas. Ninguna otra parte del trabajo afecta tanto el sistema si es hecha mal. Ninguna es tan difícil de corregir más adelante… Entonces, la tarea más importante que el ingeniero de software hace para el cliente es la extracción iterativa y el refinamiento de los requerimientos del producto.*" [1]

En la actualidad son muchos los procesos de desarrollo de software (PDSW) que existen. Con el pasar de los años, la Ingeniería de Software (ISW) ha introducido y popularizado una serie de estándares para medir y certificar la calidad, tanto del sistema a desarrollar, como del proceso de desarrollo en sí. Se han publicado muchos libros y artículos relacionados con este tema, con el modelado de procesos del negocio y la reingeniería. Un número creciente de herramientas automatizadas han surgido para ayudar a definir y aplicar un proceso de desarrollo de software efectivo. Actualmente la economía global depende más de sistemas automatizados que en épocas pasadas; esto ha llevado a los equipos de desarrollo a enfrentarse con una nueva década de procesos y estándares de calidad.

Sin embargo, ¿cómo se explica la alta incidencia de fallos en los proyectos de software? ¿Por qué existen tantos proyectos de software víctimas de retrasos, presupuestos sobregirados y con problemas de calidad? ¿Cómo se puede tener una producción o una economía de calidad, cuando nuestras actividades diarias dependen de la correctitud del sistema?

Tal vez suene ilógico pero, a pesar de los avances que ha dado la tecnología, aún existen procesos de producción informales, parciales y en algunos casos no confiables.

La Ingeniería de Requerimientos (IR) cumple un papel primordial en el proceso de producción de software, ya que enfoca un área fundamental: la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema y de esta manera, minimizar los problemas relacionados al desarrollo de los mismos. Además se ha convertido, durante los últimos años, en una de las áreas más activas en ingeniería del software. Ello es lógico considerar que la IR tiene como objetivo establecer los principios y métodos para la identificación de las restricciones de un sistema software.

Producto a lo anteriormente planteado y porque los requisitos son volátiles, mutantes o

cambiantes, emergente, colaterales y compatibles [2] es necesario investigar acerca de los procesos de ingeniería de requisitos, con el objetivo de mejorar la gestión de los mismos en las distribuciones de GNU/Linux, para contribuir a un avance positivo en la calidad del producto final.

Nova es un sistema operativo (SO) desarrollado por estudiantes y profesores de la Universidad de las Ciencias Informáticas (UCI) pertenecientes al proyecto del mismo nombre, con la participación de miembros de otras instituciones, para apoyar la migración a tecnologías de Software Libre y Código Abierto que experimenta Cuba como parte del proceso de Informatización de la Sociedad. Permite realizar trabajos de oficina, reproducir archivos de música y vídeo, navegar por Internet, ver fotografías y utilizar múltiples aplicaciones útiles para su desempeño laboral y momentos de ocio. Provee un entorno de trabajo cómodo, enfocado al usuario final, garantizando una interacción intuitiva que persigue minimizar el cambio brusco al que se enfrentan las personas familiarizadas con sistemas Microsoft Windows. Su objetivo principal así como su misión es proveer una línea de productos y servicios de calidad orientados a usuarios inexpertos en el área de las tecnologías de SWL o que hayan experimentado un proceso de migración a las mismas. Va a responder a las necesidades de las instituciones cubanas y respetar como valores fundamentales el logro de un estado de soberanía e independencia tecnológica.

En noviembre del 2010 fue realizada la segunda auditoría interna al proyecto en la que se pudo constatar que los procesos de ingeniería de requisitos (PIR) que se realizan en cada una de las líneas de desarrollo son insuficientes porque se capturan pobremente las especificaciones que va a tener la versión del producto a desarrollar, para esto sólo se basan en la lista de reserva del producto (LRP) propuesta por una distribución similar Ubuntu, por lo que no se tiene en cuenta las competencias de otros software similares, pudiendo aportar esto más calidad a la distribución a construir. Además no tienen en cuenta procesos importantes como la administración de los cambios que puedan sufrir los requisitos, el rastreo o trazabilidad de los mismos, así como la ayuda de herramientas automatizadas que permitan de una manera óptima el control de los requerimientos a cumplir. Todo esto es consecuencia de que no existan PIR enfocados al desarrollo de distribuciones GNU/Linux que estén accesibles para todos.

Esta **situación problémica** permite plantearse el siguiente **problema científico:** ¿Cómo llevar a cabo la ingeniería de requisitos en el proceso de desarrollo de distribuciones GNU/Linux?

La presente investigación tiene como **objetivo general**: Elaborar un conjunto de procesos que describan la forma de llevar a cabo la ingeniería de requisitos en el desarrollo de distribuciones de GNU/Linux.

Con el propósito de darle cumplimiento al mismo se trazan los siguientes **objetivos específicos:**

- Investigar los procesos que componen la ingeniería de requisitos así como su

comportamiento en el desarrollo de las distribuciones de GNU/Linux.

- Diseñar los procesos de ingeniería de requisitos que se deben llevar a cabo en el desarrollo de distribuciones de GNU/Linux.
- Evaluar los principales procesos propuestos.

Con el propósito de dar validez a lo anteriormente planteado se formulan las siguientes **tareas de investigación**:

- Investigación de las principales tendencias existentes a nivel mundial, en Cuba y en la UCI para saber cuáles son las prácticas que lleva a cabo la IR.
- Identificación de los procesos de ingeniería de requisitos para determinar cuáles deben ser puestos en práctica en el ciclo de vida de las distribuciones de GNU/Linux, identificando mejoras potenciales en la ejecución de estos.
- Investigación del proceso de desarrollo de software de Nova para realizar una correcta gestión de los requisitos en el mismo.
- Diseño de los PIR que se deben llevar a cabo en las distribuciones de GNU/Linux para un correcto tratamiento de las especificaciones a cumplir.
- de los PIR para las distribuciones de GNU/Linux.
- Evaluación de los principales procesos propuestos para determinar las ventajas, desventajas, y corroborar los beneficios de los mismos.

Para enmarcar los límites de esta investigación se define como **objeto de estudio:** procesos de ingeniería de requisitos, delimitando el **campo de acción** en los procesos de ingeniería de requisitos en el desarrollo de distribuciones de GNU/Linux. La investigación queda sustentada en la siguiente **idea a defender**: la utilización de un conjunto de procesos para llevar a cabo la ingeniería de requisitos en el desarrollo de las distribuciones de GNU/Linux puede mejorar la calidad del software.

Marco Metodológico

a. Definición de la Población y la Unidad de Estudio

Población

Distribuciones de GNU/Linux.

Unidad de Estudio

Distribuciones de GNU/Linux Nova.

b. Herramientas y Técnicas a utilizar

Para la realización de esta investigación se utilizaron diferentes métodos científicos, ellos

constituyen un conjunto de reglas que señalan el procedimiento para llevar a cabo una investigación.

Los métodos teóricos a utilizar son:

Analítico – Sintético: Permite estudiar de modo general el tema planteado para luego descomponerlo e interiorizar cada aspecto estudiado, sintetizando lo esencial en función de lo investigado. En dicha investigación se hace un análisis profundo acerca de cada uno de los procesos de ingeniería de requisitos centrándose en los aportes de los mismos para las distribuciones de GNU/Linux.

Análisis histórico-lógico: se encarga de investigar los antecedentes de los elementos en el transcurso del tiempo, no solo se limita a la descripción del hecho, sino que analiza la lógica de su desarrollo. En este trabajo se hace un estudio de los orígenes y evolución de la IR para establecer procesos que guíen a mejorar la gestión de requisitos (GR) de las distribuciones de GNU/Linux.

El método empírico a utilizar es:

Entrevista: Este método se realiza a través de diálogos, con el fin de obtener una información crucial para la investigación. En este proceso se le hace entrevistas a un grupo de líderes de las líneas de desarrollo del proyecto Nova, para conocer las características de cada producto desarrollado así como para determinar de qué manera se ejecutan los PIR para perfeccionar los existentes, teniendo en cuenta las actividades para desarrollar distribuciones de GNU/Linux. Luego de establecer el diseño de los procesos se realizan entrevistas a especialistas en el tema para evaluar los mismos.

Resultados esperados

Conjunto de procesos que describen la forma de llevar a cabo la ingeniería de requisitos en el desarrollo de las distribuciones de GNU/Linux.

Estructuración de los capítulos

Capítulo 1: Fundamentos teóricos de la ingeniería de requisitos.

Capítulo 2: Procesos de ingeniería de requisitos en el desarrollo de las distribuciones de GNU/Linux.

Capítulo 3: Evaluación de los procesos de IR propuestos.

Capítulo 1: Fundamentos teóricos de la IR

En el presente capítulo se hace un análisis del surgimiento de la IR, las principales tendencias y la descripción de los principales conceptos asociados. Se analizan los procesos de IR, se presentan las principales ventajas y desventajas de cada una de las técnicas utilizadas en las etapas de la IR y en base a estas se realiza una comparación entre algunas de las técnicas presentadas. Además se estudia el proceso de desarrollo de software de distribuciones GNU/Linux.

1.1. Principales conceptos que se tienen en cuenta en la IR

1.1.1. Requerimientos

- Una condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo.
- Una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, una especificación u otro documento formal.
- Una representación en forma de documento de una condición o capacidad como las expresadas anteriormente.

  Los requisitos pueden dividirse en funcionales y no funcionales. Los funcionales definen las funciones que el sistema va a ser capaz de realizar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas y los no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema. **[2]**

1.1.2. Ingeniería de Requisitos (IR)

- La IR ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajan. Incluye el conjunto de tareas que conducen a comprender cuál es el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactúan los usuarios finales con el software.
- La IR es el proceso de desarrollar una especificación de software. Las especificaciones pretenden comunicar las necesidades del sistema del cliente a los desarrolladores del sistema.
- La IR se define, como un conjunto de actividades en las cuales, utilizando técnicas y herramientas, se analiza un problema y se concluye con la especificación de una solución (a veces más de una). **[1]**

1.1.3. Proceso de ingeniería de requerimiento (PIR)

El proceso de recopilar, analizar y verificar las necesidades del cliente para un sistema es llamado Ingeniería de Requerimientos. La meta de la IR es entregar una especificación de requisitos de software correcta y completa. **[3]**

1.1.4. Trazabilidad de Requisitos

Según la IEEE 830-1998 es la habilidad para seguir la vida de un requisito en ambos sentidos, hacia sus orígenes o hacia su implementación, a través de las especificaciones generadas durante el proceso de desarrollo. **[4]**

1.1.5. Repositorio

Es un término utilizado en el dominio de las herramientas CASE (Computer Aided Software Engineering). El repositorio podría definirse como la base de datos fundamental para el diseño; no sólo guarda datos, sino también algoritmos de diseño y, en general, elementos software necesarios para el trabajo de programación. **[5]**

1.1.6. Compilación de Software.

El programa escrito en un lenguaje de programación no es inmediatamente ejecutado en una computadora. La opción más común es compilar el programa, aunque también puede ser ejecutado mediante un intérprete informático. El código fuente del programa se debe someter a un proceso de transformación para convertirse en lenguaje máquina, interpretable por el procesador. A este proceso se le llama compilación.

1.1.7. Pruebas de Software

Es el conjunto de técnicas que permiten determinar la calidad de un producto software. Las Pruebas de software se integran dentro de las diferentes fases del ciclo del software. Así se ejecuta un programa y mediante técnicas experimentales se trata de descubrir que errores tiene. **[6]**

1.2. ¿Cuáles son las tendencias de la IR?

La Ingeniería como concepto y como palabra es tan antigua como la existencia del hombre en la tierra, proviene del vocablo latino INGENIUM que significa cualidades innatas. **[7]**

Durante los primeros años de la informática, el software se consideraba como un añadido. En una segunda época (a partir de mitad de la década de 1960) se estableció el software como producto y

aparecieron las empresas dedicadas al desarrollo y distribución masiva del mismo.

Actualmente se considera la ISW como una nueva área de la ingeniería, y la profesión de ingeniero informático es una de las más demandadas. El ciclo de vida del desarrollo del software por lo general incluye diversas fases entre las que se encuentra la fase de requisitos por lo que surge la IR como una necesidad de contar con un enfoque metódico, sistemático y disciplinado que le permita a los desarrolladores resolver uno de los problemas principales que afecta al éxito de los proyectos, una mala gestión de requerimientos, por fallas en la adquisición, evaluación y/o documentación de los mismos así como dificultades en la comunicación y/o cambios no controlados durante el proyecto.

La IR es una de las partes más importantes y menos apreciadas de la ISW. Como todas las ramas de la ingeniería relativas a la fabricación de software su origen es muy reciente y es hasta hace poco tiempo que es reconocida por los principales autores como una disciplina formal.

El auge que esta disciplina ha tenido en los últimos años se debe a la importante influencia que los requerimientos muestran en las estadísticas de proyectos fracasados o con problemas de tiempo o presupuesto. **[8]**

1.2.1. Características de los requerimientos

Un conjunto de requerimientos en estado de madurez, deben presentar una serie de características tanto individualmente como en grupo. Las más importantes son:

- **Necesario:** Un requerimiento es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.
- **Conciso:** Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- **Completo:** Un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.
- **Consistente:** Un requerimiento es consistente si no es contradictorio con otro requerimiento.
- **No ambiguo:** Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.
- **Verificable:** Un requerimiento es verificable cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración o pruebas.

1.2.2. Clasificación de requisitos

La prioridad de cada requerimiento dependerá de las necesidades que tenga el negocio. En base a la prioridad, cada requerimiento puede ser clasificado como:

- **Mandatorio:** Afecta una operación crítica del negocio.
- **Deseables:** Si existe algún proceso que se quiera incluir para mejorar los procesos actuales.
- **Innecesarios:** Se trata de un requerimiento informativo o que puede esperar para fases posteriores. **[2]**

1.2.3. Control de cambios a los requerimientos

El control de cambios es el proceso mediante el cual se asegura que no se realicen cambios que afecten el éxito del proyecto, y que aquellos que se implementen sean analizados, negociados y planeados de una manera adecuada.
Lo único constante en los proyectos son los cambios. Se debe acostumbrar a ellos y aceptarlos como algo normal. Cambios que solicitan los usuarios porque comprenden mejor lo que necesitan, o porque cambian las necesidades del negocio, porque se identifica una mejor forma de hacer las cosas o por cualquier otra razón.

El problema no son los cambios a los requerimientos, sino el hecho de que se agreguen a la lista de nuevas especificaciones del proyecto sin considerar el impacto que tienen sobre el plan. No tener esto en cuenta significa que cuando el proyecto se termine en una fecha posterior a la acordada originalmente, o con un presupuesto mayor al considerado, se le podría culpar al líder del proyecto como un fracaso.

El cambio es analizado y se evalúa el impacto en costo y tiempo, y si es algo aceptable para los recursos disponibles y el tiempo que se le puede asignar a dicho proyecto, además de ser aceptado por el usuario y autorizado por la gerencia, entonces se acepta la solicitud. En caso contrario debe registrarse como una solicitud rechazada.

Independientemente de que la solicitud sea aceptada o rechazada debe registrarse en el control de cambios del proyecto con un identificador único y algunos datos básicos de acuerdo al formato establecido para ello, o de acuerdo a la herramienta de control de cambios que se utilice. **[9]**

1.2.4. Trazabilidad de requisitos

La trazabilidad de requisitos es considerada un proceso imprescindible para la adecuada gestión de requisitos. El principal problema de dicho proceso es la falta de consenso en cuanto a los tipos de enlace de trazabilidad que han de considerarse, la semántica de cada uno de ellos y los tipos de artefactos entre los que se establecen dichos enlaces. Otro factor clave a tener en cuenta es la adaptabilidad de la trazabilidad a las necesidades específicas de un proyecto. **[10]**

1.2.5. Importancia de la IR

Los principales beneficios que se obtienen de la IR son:

- **Permite gestionar las necesidades del proyecto en forma estructurada:** Cada actividad de la IR consiste de una serie de pasos organizados y bien definidos.
- **Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados:** La IR proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios.
- **Disminuye los costos y retrasos del proyecto:** Muchos estudios han demostrado que reparar errores por un mal desarrollo no descubierto a tiempo, es sumamente caro.
- **Mejora la calidad del software:** La calidad en el software tiene que ver con cumplir un conjunto de factores (funcionalidad, facilidad de uso, confiabilidad, desempeño, etc.).
- **Mejora la comunicación entre equipos:** La especificación de requerimientos representa una forma de consenso entre clientes y desarrolladores. Si este consenso no ocurre, el proyecto no será exitoso.
- **Evita rechazos de usuarios finales:** La ingeniería de requerimientos obliga al cliente a considerar sus requerimientos cuidadosamente y revisarlos dentro del marco del problema, por lo que se le involucra durante todo el desarrollo del proyecto.

1.2.6. Personal involucrado en la IR

Realmente, son muchas las personas involucradas en el desarrollo de los requerimientos de un sistema. Es importante saber que cada una de esas personas tienen diversos intereses y juegan roles específicos dentro de la planificación del proyecto; el conocimiento de cada papel desempeñado, asegura que se involucren a las personas correctas en las

diferentes fases del ciclo de vida, y en las diferentes actividades de la IR.

No conocer estos intereses puede ocasionar una comunicación poco efectiva entre clientes y desarrolladores, que a la vez traería impactos negativos tanto en tiempo como en presupuesto. Los roles más importantes pueden clasificarse como sigue:

- **Usuario final:** Son las personas que usan el sistema desarrollado. Ellos están relacionados con la usabilidad, la disponibilidad y la fiabilidad del sistema; están familiarizados con los procesos específicos que debe realizar el software, dentro de los parámetros de su ambiente laboral. Son quienes utilizan las interfaces y los manuales de usuario. Además son las personas que van a optimizar su trabajo con la asistencia del software.
- **Usuario líder:** Son los individuos que comprenden el ambiente del sistema o el dominio del problema donde es empleado el software desarrollado. Ellos proporcionan al equipo técnico los detalles y requerimientos de las interfaces del sistema.
- **Personal de Mantenimiento:** Para proyectos que requieran un mantenimiento eventual, estas personas son las responsables de la administración de cambios, de la implementación y resolución de anomalías. Su trabajo consiste en revisar y mejorar los procesos del producto ya finalizado.
- **Analistas y programadores:** Son los responsables del desarrollo del producto en sí; ellos interactúan directamente con el cliente.

  Competencias del analista

  - Actitud crítica, de perfeccionamiento y actualización permanente
  - Capacidad para trabajar en forma cooperativa y constructiva, en equipos multidisciplinarios
  - Capacidad de abstracción
  - Permanente capacidad para derivar de la práctica elaboraciones conceptuales.
  - Orientación al cliente
  - Pro actividad
  - Flexibilidad
  - Capacidad de análisis y solución de problemas
  - Escucha y comunicación

  Competencias de los programadores

  - Capacidad de aprendizaje

- Trabajo en equipo
- Pro actividad
- Flexibilidad
- Capacidad de análisis y solución de problemas
- Orientación al cliente
- Capacidad para establecer prioridades
- Escucha y comunicación
- Orientación a resultados
- Pensamiento de integración
- Capacidad para detectar riesgos

- **Personal de pruebas:** Se encargan de elaborar y ejecutar el plan de pruebas para asegurar que las condiciones presentadas por el sistema son las adecuadas. Son quienes evalúan si los requerimientos satisfacen las necesidades del cliente.

Otras personas que pueden estar involucradas, dependiendo de la magnitud del proyecto, pueden ser: administradores de proyecto, documentadores, diseñadores de base de datos, entre otros. **[11]**

1.3. Modelos, estándares, normas que contemplan la IR

MODELO CMM NIVEL REPETITIVO (2)

El nivel 2 de CMMI pese al ser el primer nivel a alcanzar es muchas veces el más difícil de alcanzar y esto es porque requiere que se cambie la forma de trabajar de la empresa, lo que la mayoría de las veces implica un cambio cultural de la misma. Por este motivo es necesario un fuerte apoyo de la dirección para afrontar este cambio, ya que sin él no tendrás suficiente autoridad en momentos difíciles, resumiendo: **No intentes alcanzar el CMM-CMMI nivel 2 sin un firme apoyo de la dirección**.

Lo que se pretende con el nivel 2 de CMM-CMMI es conseguir que en los proyectos de la organización haya una gestión de los requisitos y que los procesos (formas de hacer las cosas) estén planeados, ejecutados, medidos y controlados.

Áreas de proceso del Nivel 2 de CMM-CMMI

- Gestión de Requisitos o Requerimientos
- Planificación de proyectos
- Medición y Análisis
- Aseguramiento de la calidad

- Gestión de la configuración **[12]**

Actividades de la Ingeniería de requisitos para este modelo

- Identificación de requerimientos
- Análisis de los requerimientos
- Representación de los requerimientos
- Comunicación de los requerimientos
- Evaluación de requerimientos **[2]**

Las normas presentadas a continuación tratan elementos relacionados con la IR.

- IEEE 610.12. Es la norma que especifica la especificación de requisitos. Se complementa con la IEEE 830-1998 para la estructura del documento de la especificación de requisitos, la cual a su vez define los beneficios de una buena especificación de requisitos de software (ERS):
  - Establecer las bases para un acuerdo entre los clientes y los proveedores en lo que el producto de software necesita hacer.
  - Reducir el esfuerzo de desarrollo.
  - Proporcionar una base para la estimación de costos y horarios
  - Proporcionar un punto de referencia para la evaluación y verificación.
  - Facilitar la transferencia.
  - Servir como base para la mejora.

¿Por qué utilizar estándares?

- Son indispensables cuando muchas personas, productos y herramientas deben coexistir.
- Promueven el buen uso de métodos y herramientas.
- Permiten la comunicación entre los desarrolladores.
- Facilitan el mantenimiento del software.
- Facilitan la capacitación del personal.
- Proveen una base para evaluar los diferentes productos de software.
- Permiten definir el proceso de software de una organización. **[13]**

1.4. Principales actividades de la IR [2]

ANÁLISIS DEL PROBLEMA

Descripción	El objetivo de esta actividad es entender las verdaderas necesidades del negocio. Además que se evalúen las necesidades iniciales de todos los involucrados en el proyecto y que se proponga una solución de alto nivel para resolverlo.
Entradas	Problema.
Personas que intervienen	Para saber quiénes son las personas, departamentos, organizaciones internas o externas que se verán afectadas por el sistema, se realizan algunas preguntas. ¿Quién usará el sistema que se va a construir? ¿Quién desarrollará el sistema? ¿Quién probará el sistema? ¿Quién documentará el sistema? ¿Quién dará soporte al sistema? ¿Quién dará mantenimiento al sistema? ¿Quién mercadeará, venderá, y/o distribuirá el sistema? ¿Quién se beneficiará por el retorno de inversión del sistema? Como ven, debe conocerse la opinión de todo aquél que de una u otra forma está involucrado con el sistema, ya sea directa o indirectamente.
Actividades	**Comprender el problema que se está resolviendo:** Es importante determinar quién tiene el problema realmente, considerar dicho problema desde una variedad de perspectivas y explorar muchas soluciones desde diferentes puntos de vista. Las soluciones iniciales, deben ser definidas tomando en cuenta tanto la perspectiva técnica como la del negocio. **Construir un vocabulario común:** Debe confeccionarse un glosario en dónde se definan todos los términos que tengan significados comunes (sinónimos) y que son utilizados durante el proyecto. **Identificar a los afectados por el sistema:** Identificar a todos los afectados evita que existan sorpresas a medida que avanza el proyecto. Las necesidades de cada afectado, son discutidas y sometidas a debate durante la IR, aunque esto no garantiza que vaya a estar disponible toda la información necesaria para especificar un sistema adecuado. **Definir los límites y restricciones del sistema:** Este punto es importante

	pues debemos saber lo que se está construyendo, y lo que no se está construyendo, para así entender la estrategia del producto a corto y largo plazo. Debe determinarse cualquier restricción ambiental, presupuestaria, de tiempo, técnica y de factibilidad que limite el sistema que se va a construir.
Salidas	Especificación de requisitos de software
Aplicación	➢ Redactar en media cuartilla el problema planteado ➢ Elaborar el vocabulario común ➢ Identificar los afectados del sistema ➢ Definir los límites y restricciones del problema a solucionar

EVALUACIÓN Y NEGOCIACIÓN DE LOS REQUERIMIENTOS	
Descripción	En esta etapa se pretende limitar las expectativas del cliente apropiadamente, tomando como referencia los niveles de abstracción y descomposición de cada problema presentado.
Entradas	Requisitos.
Personas que intervienen	Analistas, programadores y usuario final.
Actividades	**Descubrir problemas potenciales:** En este paso se asegura que todas las características estén presentes en cada uno de los requerimientos, es decir, se identifican aquellos requerimientos ambiguos, incompletos, inconsistentes, etc. **Clasificar los requerimientos:** En este paso se busca identificar la importancia que tiene un requerimiento en términos de implementación. A esta característica se le conoce como prioridad y debe ser usada para establecer la secuencia en que ocurren las actividades de diseño y prueba de cada requisito. La prioridad de cada requerimiento depende de las necesidades que tenga el negocio. En base a la prioridad, cada requerimiento puede ser clasificados como mandatorio, deseables o

	innecesarios. Una vez hecha esta categorización de los requerimientos, puedo tomar como estrategia general el incluir los mandatorios, discutir los deseables y descartar los innecesarios. Antes de decidir la inclusión de un requerimiento, también debe analizarse su costo, complejidad, y una cantidad de otros factores. Por ejemplo, si un requerimiento fuera trivial de implementar, puede ser una buena idea incluirlo por más que éste sea sólo deseable. **Evaluar factibilidades y riesgos:** Involucra la evaluación de factibilidades técnicas (¿pueden implementarse los requerimientos con la tecnología actual?); factibilidades operacionales (¿puede ser el sistema utilizado sin alterar el organigrama actual?); factibilidades económicas (¿ha sido aprobado por los clientes el presupuesto?). En Esta actividad se incrementa la comunicación entre el equipo de desarrollo y los afectados. Para que los requerimientos puedan ser comunicados de manera efectiva, hay una serie de consideraciones que deben tenerse en cuenta; entre las principales tenemos: ➢ Documentar todos los requerimientos a un nivel de detalle apropiado. ➢ Mostrar todos los requerimientos a los involucrados en el sistema. ➢ Analizar el impacto que causen los cambios a requerimientos antes de aceptarlos. ➢ Establecer las relaciones entre requerimientos que indiquen dependencias. ➢ Negociar con flexibilidad para que exista un beneficio mutuo. ➢ Enfocarse en intereses y no en posiciones.
Salidas	Especificación de requisitos de software
Aplicación	➢ Entregar documentos en donde se enlisten los requerimientos del sistema planteando los puntos vistos anteriormente, dicho documento será la carta de presentación de los equipos. ➢ Exponer ante los compañeros los requerimientos fundamentales para llevar a buen término la solución del problema a plantear.

ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE (ERS)	
Descripción	La especificación de requisitos de software es la actividad en la cual se genera el documento, con el mismo nombre, que contiene una descripción completa de las necesidades y funcionalidades del sistema que será desarrollado; describe el alcance del sistema y la forma en cómo hará sus funciones, definiendo los requerimientos funcionales y los no funcionales. La ERS posee las mismas características de los requerimientos: completa, consistente, verificable, no ambigua, factible, modificable, rastreable, precisa, entre otras. Para que cada característica de la ERS sea considerada, cada uno de los requerimientos debe cumplirlas. La estandarización de la ERS es fundamental pues ayudará, entre otras cosas, a facilitar la lectura y escritura de la misma. Será un documento familiar para todos los involucrados, además de asegurar que se cubren todos los tópicos importantes.
Entradas	Problema y los requisitos.
Personas que intervienen	Clientes, usuarios finales, analistas de sistema, personal de pruebas, y todo aquel involucrado en la implementación del sistema.
Actividades	En la ERS se definen todos los requerimientos de hardware y software, diagramas, modelos de sistemas y cualquier otra información que sirva de soporte y guía para fases posteriores. Es importante destacar que la especificación de requisitos es el resultado final de las actividades de análisis y evaluación de requerimientos. Este documento resultante será utilizado como fuente básica de comunicación entre las personas que intervienen.
Salidas	Documento de ERS.
Aplicación	Los clientes y usuarios utilizan la ERS para comparar si lo que se está proponiendo, coincide con las necesidades de la empresa. Los analistas y programadores la utilizan para determinar el producto que debe desarrollarse.

	El personal de pruebas elaborará las pruebas funcionales y de sistemas en base a este documento. Para el administrador del proyecto sirve como referencia y control de la evolución del sistema.

VALIDACIÓN DE REQUISITOS	
Descripción	La evaluación es la actividad de la IR que permite demostrar que los requerimientos definidos en el sistema son los que realmente quiere el cliente; además revisa que no se haya omitido ninguno, que no sean ambiguos, inconsistentes o redundantes. En este punto es necesario recordar que la ERS debe estar libre de errores, por lo tanto, la evaluación garantiza que todos los requerimientos presentes en el documento de especificación sigan los estándares de calidad. No debe confundirse la actividad de evaluación de requerimientos con la validación de requerimientos. La evaluación verifica las propiedades de cada requerimiento, mientras que la validación revisa el cumplimiento de las características de la especificación de requisitos.
Entradas	Documento de ERS.
Personas que intervienen	Personal de pruebas
Actividades	Durante la actividad de evaluación pueden hacerse preguntas en base a cada una de las características que se desean revisar. A continuación se presentan varios ejemplos: ➢ ¿Están incluidas todas las funciones requeridas por el cliente? (completa) ➢ ¿Existen conflictos en los requerimientos? (consistencia) ➢ ¿Tiene alguno de los requerimientos más de una interpretación? (no ambigua) ➢ ¿Está cada requerimiento claramente representado? (entendible) ➢ ¿Pueden los requerimientos ser implementados con la tecnología y el presupuesto disponible? (factible)

	➢ ¿Está la ERS escrita en un lenguaje apropiado? (clara) ➢ ¿Existe facilidad para hacer cambios en los requerimientos? (modificable) ➢ ¿Está claramente definido el origen de cada requisito? (rastreable) ➢ ¿Pueden los requerimientos ser sometidos a medidas cuantitativas? (verificable)
Salidas	Plantilla No Conformidades Acciones correctivas Producto modificado
Aplicación	Se elabora y ejecuta un plan de pruebas para asegurar que las condiciones presentadas por el sistema son las adecuadas.

EVOLUCIÓN DE LOS REQUERIMIENTOS	
Descripción	La actividad de evolución es un proceso externo que ocurre a lo largo del ciclo de vida del proyecto. Cambios a los requisitos involucra modificar el tiempo en el que se va a implementar una característica en particular, modificación que a la vez puede tener impacto en otros requerimientos. Los requerimientos cambian por diferentes razones. Las más frecuentes son: ➢ Porque al analizar el problema, no se hacen las preguntas correctas a las personas correctas. ➢ Porque cambió el problema que se estaba resolviendo. ➢ Porque los usuarios cambiaron su forma de pensar o sus percepciones. ➢ Porque cambió el ambiente de negocios. ➢ Porque cambió el mercado en el cual se desenvuelve el negocio.
Entradas	Documento de ERS revisado.
Personas que intervienen	Clientes, usuarios finales, analistas de sistema, personal de pruebas, y todo aquel involucrado en la

	implementación del sistema.
Actividades	Establecer políticas, guardar históricos de cada requerimiento, identificar dependencias entre ellos y mantener un control de versiones.
Salidas	Tener versiones de los requerimientos es tan importante como tener versiones del código, ya que evita tener requerimientos emparchados en un proyecto Entre algunos de los beneficios que proporciona el control de versiones están: ➢ Prevenir cambios no autorizados. ➢ Guardar revisiones de los documentos de requerimientos. ➢ Recuperar versiones previas de los documentos. ➢ Administrar una estrategia de "releases". ➢ Prevenir la modificación simultánea a los requisitos. Producto modificado.
Aplicación	En vista que las peticiones de cambios provienen de muchas fuentes, las mismas deben ser enrutadas en un solo proceso. Esto se hace con la finalidad de evitar problemas y conseguir estabilidad en los requerimientos.

1.5. Metodologías de desarrollo de software

1.5.1. RUP (Proceso Racional Unificado)

Este es uno de los procesos más generales que existe, está enfocado a cualquier tipo de proyecto así no sea de software.

1. **Concepción:** Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos potenciales asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones.
2. **Elaboración:** En la fase de elaboración se seleccionan los casos de uso que permiten definir la arquitectura base del sistema y se desarrollaran en esta fase, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar.
3. **Construcción:** El propósito de esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requerimientos pendientes, administrar los cambios de acuerdo a las evaluaciones realizados por los usuarios y se realizan las mejoras para el proyecto.
4. **Transición:** El propósito de esta fase es asegurar que el software esté disponible para los usuarios finales, ajustar los errores y defectos encontrados en las pruebas de aceptación,

capacitar a los usuarios y proveer el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto.

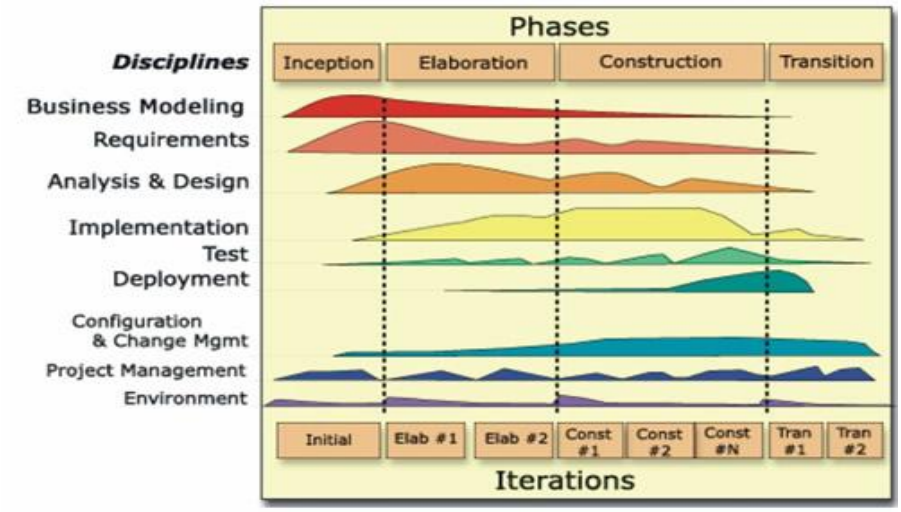


Figura 1: Fases de RUP

RUP es dirigida por casos de uso para describir lo que se tiene y lo que se espera del software, está muy orientado a la arquitectura del sistema a implementarse, documentándose de la mejor manera, basándose en UML (Unified Modeling Language - Lenguaje de Modelado Unificado).

Para poder usar RUP antes hay que adaptarlo a las características de la organización, y medir de manera exacta el tiempo, costos y todos los demás recursos involucrados en el proceso.

Gestión de requisitos

RUP brinda una guía que se dedica a encontrar, organizar, documentar, y seguir los cambios de los requisitos. Utiliza una notación de Caso de Uso y escenarios para representar las funcionalidades a cumplir**. [14]**

Requisitos

Este es uno de los flujos de trabajo más importantes, porque en él se establece qué tiene que hacer exactamente el sistema que construye. En esta línea los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que especifican.

Los objetivos del flujo de datos Requisitos son:

- Establecer y mantener un acuerdo entre clientes y otros stakeholders sobre lo que el sistema puede hacer.
- Proveer a los desarrolladores un mejor entendimiento de los requisitos del sistema.
- Definir el ámbito del sistema.

- Proveer una base para la planeación de los contenidos técnicos de las iteraciones.
- Proveer una base para estimar costos y tiempo de desarrollo del sistema.
- Definir una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario.

Para capturar los requisitos es preciso entrevistar a todos los interesados en el proyecto, no sólo a los usuarios finales, y anotar todas sus peticiones. A partir de ellas hay que descubrir lo que necesitan y expresarlo en forma de requisitos.

En este flujo de trabajo, y como parte de los requisitos de facilidad de uso, se diseña la interfaz gráfica de usuario. Para ello habitualmente se construyen prototipos de la interfaz gráfica de usuario que se contrastan con el usuario final.

Administración de requisitos

RUP describe cómo:

- Obtener los requisitos
- Organizarlos
- Documentar requisitos de funcionalidad y restricciones
- Rastrear y documentar decisiones
- Captar y comunicar requisitos del negocio **[15]**

Actividades de IR en RUP

- Análisis del Problema
- Comprender las necesidades de los involucrados
- Definir el sistema
- Analizar el alcance del proyecto
- Modificar la definición del sistema
- Administrar los cambios de requerimientos **[2]**

Ventajas y desventajas

Incluye prácticas claves y aspectos relacionados a la planeación estratégica y administración de riesgos; y actualmente guían de forma natural el proceso de desarrollo de software complejo por lo que ha sido considerado como un estándar el desarrollo de software en las empresas.

El proceso unificado RUP, es un modelo de software que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo. Sin embargo, los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto.

El proceso de desarrollo constituye un marco metodológico que define en términos de metas estratégicas, objetivos, actividades y artefactos (documentación) requerido en cada fase de desarrollo. Esto permite enfocar esfuerzo de los recursos humanos en términos de habilidades, competencias y capacidades a asumir roles específicos con responsabilidades bien definidas. **[14]**

1.5.2. OpenUP (Proceso Unificado Abierto)

OpenUP es una metodología open source que forma parte del Eclipse Foundation. Es derivada de RUP, pero fue simplificada para ser transformada en una metodología más ágil para proyectos de desarrollo de software. **[16]**

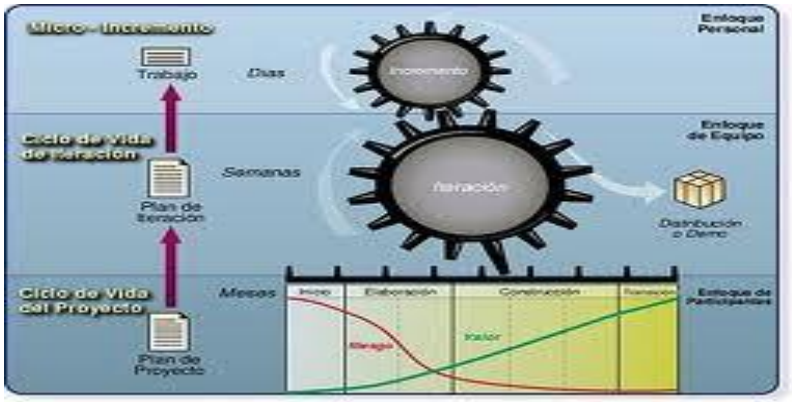


Figura 2: Ciclo de vida de OpenUP

OpenUP es un proceso iterativo para el desarrollo de software que es:

- **Mínimo:** Solo incluye el contenido del proceso fundamental.
- **Completo:** Puede ser manifestado como proceso entero para construir un sistema.
- **Extensible:** Puede ser utilizado como base para agregar o para adaptar más procesos.

Beneficios en el uso del OpenUP

- Ya que es apropiado para proyectos pequeños y de bajos recursos permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito.
- Permite detectar errores tempranos a través de un ciclo iterativo.
- Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP.

- Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas. **[17]**

OpenUP conserva las características principales del modelo de desarrollo RUP, incluye el desarrollo iterativo, permite identificar los requisitos operacionales del sistema, proveer las interacciones con los usuarios y prevenir los posibles riesgos en el desarrollo del sistema, es una forma de desarrollo más ágil y ligera, consiste en equipos a los cuales se les asigna una fase del desarrollo que tienen que complementarse entre sí para obtener un buen producto final, no puede ser una sola persona la que realice todo el trabajo pues esto podría ocasionar que se pierda de vista ciertas características importantes, por ejemplo para un proyecto pequeño constituyen equipos de 3 a 6 personas e implican 3 a 6 meses de esfuerzo del desarrollo.

Los requisitos del usuario se priorizan y los requisitos de prioridad más alta se incluyen en los incrementos más tempranos. Cuando el desarrollo de un incremento comienza, sus requisitos son inamovibles, aunque los requisitos de incrementos posteriores pueden continuar desarrollándose. Los clientes no tienen que esperar hasta tener el sistema completo. El primer incremento satisface los requisitos más críticos. Los primeros incrementos sirven como prototipo y ayudan en la tarea de detectar los posteriores requisitos**. [16]**

OpenUP mantiene las características esenciales de RUP, en el cual se incluyen las siguientes características:

1. Desarrollo incremental.
2. Uso de casos de uso y escenarios.
3. Manejo de riesgos.
4. Diseño basado en la arquitectura.

Su proceso puede ser personalizado y extendido para distintas necesidades, que aparecen a lo largo del ciclo de vida del desarrollo de software, dado que su modelo de desarrollo es incremental iterativo, es capaz de producir versiones, además, una de sus mayores ventajas es que puede ser acoplado para proyectos pequeños, dado que en su gráfica de roles aparecen 4 personas, que pueden trabajar bien manejando esta metodología.

Dado que mantiene las bases de RUP, aún maneja procesos tan importantes como Manejo de Riesgos, que es una parte del desarrollo de software que no se puede descuidar, una desventaja es que se puede utilizar este modelo sin tanto formalismo y podemos caer en el desorden y perder la trazabilidad del proyecto. **[18]**

1.5.3. XP (Programación Extrema)

- Es un proceso ligero, ágil, de bajo riesgo, flexible, predecible, científico y divertido de desarrollar software.
- Está orientado hacia quien produce y usa el software (retroalimentación continua cliente y desarrollador).
- Reduce el costo del cambio en todas las etapas del ciclo de vida del sistema.
- Combina las que han demostrado ser las mejores prácticas para desarrollar software, y las lleva al extremo.
- Metodología creada a base de prueba y error.
- Énfasis en el desarrollo del software más que una buena documentación.
- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- No introduce funcionalidades antes de que sean necesarias.
- El cliente o el usuario se convierten en miembro del mismo equipo.

Su utilidad es medida con cuatro valores:

- Simplicidad en las soluciones implementadas.
- Comunicación.
- Retroalimentación.
- Coraje

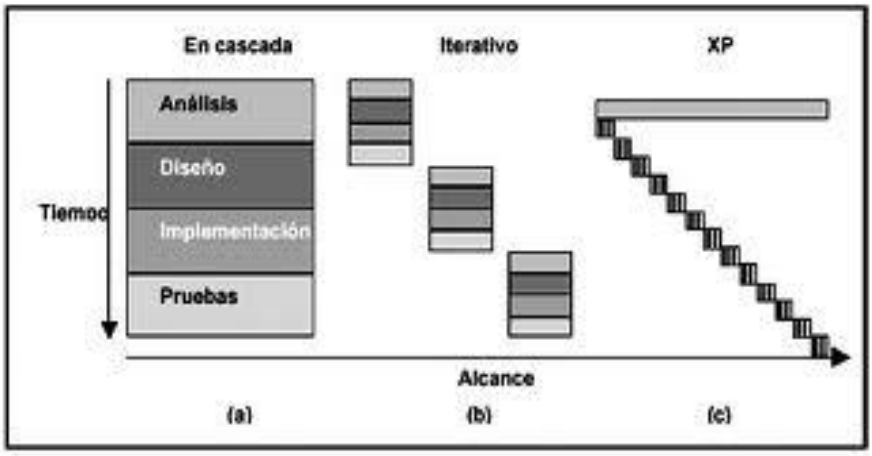


Figura 3: Fases de XP

Captura de Requisitos en XP

Historias del Usuario (User-Stories)

- Establecen los requisitos del cliente
- Trozos de funcionalidad que aportan valor
- Se les asignan tareas de programación con un número de horas de desarrollo

- Las establece el cliente

Ventajas:

- Programación organizada.
- Menor taza de errores.
- Satisfacción del programador.

Desventajas:

- Es recomendable emplearlo solo en proyectos a corto plazo.
- Altas comisiones en caso de fallar. **[19]**

1.6. Técnicas y Herramientas utilizadas en las actividades de IR

Existen varias técnicas para la IR. Es importante resaltar que estas técnicas pueden ser aplicables a las distintas fases de este proceso, haciendo la salvedad de que hay que tomar en cuenta las características propias del proyecto que esté desarrollándose para aprovechar al máximo su utilidad.

- **Entrevistas y cuestionarios**

  Las entrevistas y cuestionarios se emplean para reunir información proveniente de personas o de grupos. Por lo común, los encuestados son usuarios de los sistemas existentes o usuarios en potencia del sistema propuesto. En algunos casos, son gerentes o empleados que proporcionan datos para el sistema propuesto o que son afectados por él. El éxito de esta técnica, depende de la habilidad del entrevistador y de su preparación para la misma.

  Durante la entrevista, el analista conversa con el encuestado; el cuestionario consiste en una serie de preguntas relacionadas con varios aspectos de un sistema.

  Ventajas

  - Mediante ellas se obtiene una gran cantidad de información correcta a través del usuario.
  - Pueden ser usadas para obtener un pantallazo del dominio del problema.
  - Son flexibles.
  - Permiten combinarse con otras técnicas.

  Desventajas

  - La información obtenida al principio puede ser redundante o incompleta.

- Si el volumen de información manejado es alto, requiere mucha organización por parte del analista, así como la habilidad para tratar y comprender el comportamiento de todos los involucrados.

➢ **Lluvia de ideas (Brainstorm)**

Este es un modelo que se usa para generar ideas. La intención en su aplicación es la de generar la máxima cantidad posible de requerimientos para el sistema. No hay que detenerse en pensar si la idea eso no del todo utilizable. La intención de este ejercicio es generar, en una primera instancia, muchas ideas.

Luego, se irán eliminando en base a distintos criterios como, por ejemplo, "caro", "impracticable", "imposible", etc.

Las reglas básicas a seguir son:

- Los participantes deben pertenecer a distintas disciplinas y, preferentemente, deben tener mucha experiencia. Esto trae aparejado la obtención de una cantidad mayor de ideas creativas.
- Conviene suspender el juicio crítico y se debe permitir la evolución de cada una de las ideas, porque si no se crea un ambiente hostil que no alienta la generación de ideas.
- Por más locas o salvajes que parezcan algunas ideas, no se las debe descartar, porque luego de maduradas probablemente se tornen en un requerimiento sumamente útil.
- A veces ocurre que una idea resulta en otra idea, y otras veces podemos relacionar varias ideas para generar una nueva.
- Escribir las ideas sin censura.

Ventajas

- Los diferentes puntos de vista y las confusiones en cuanto a terminología, son aclaradas por especialistas.
- Ayuda a desarrollar ideas unificadas basadas en la experiencia de un especialista.

Desventajas

- Es necesaria una buena compenetración del grupo participante.

➢ **Prototipos**

Los prototipos son simulaciones del posible producto, que luego son utilizados por el usuario final, permitiendo conseguir una importante retroalimentación en cuanto a si el

sistema diseñado con base a los requerimientos recolectados le permite al usuario realizar su trabajo de manera eficiente y efectiva.

Durante la actividad de extracción de requerimientos, puede ocurrir que algunos requerimientos no estén demasiado claros o que no se esté muy seguro de haber entendido correctamente los requerimientos obtenidos hasta el momento, todo lo cual puede llevar a un desarrollo no eficaz del sistema final.

Entonces, para evaluar los requerimientos hallados, se construyen prototipos. El desarrollo del prototipo comienza con la captura de requerimientos. Desarrolladores y clientes se reúnen y definen los objetivos globales del software, identifican todos los requerimientos que son conocidos, y señalan áreas en las que será necesaria la profundización en las definiciones. Luego de esto, tiene lugar un "diseño rápido". El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles al usuario (por ejemplo, entradas y formatos de las salidas). El diseño rápido lleva a la construcción de un prototipo.

Ventajas

- Ayudan a evaluar y desarrollar nuevos requerimientos.
- Permite comprender aquellos requerimientos que no están muy claros y que son de alta volatilidad.

Desventajas

- El cliente puede llegar a pensar que el prototipo es una versión del software que será desarrollado.
- A menudo, el desarrollador hace compromisos de implementación con el objetivo de acelerar la puesta en funcionamiento del prototipo.

- **Proceso de Análisis Jerárquico**

Esta técnica tiene por objetivo resolver problemas cuantitativos, para facilitar el pensamiento analítico y las métricas. Consiste en una serie de pasos a saber:

- Encontrar los requerimientos que van a ser priorizados.
- Combinar los requerimientos en las filas y columnas de la matriz n x n de AHP.
- Hacer algunas comparaciones de los requerimientos en la matriz

- Sumar las columnas
- Normalizar la suma de las filas
- Calcular los promedios

Estos pasos pueden aplicarse fácilmente a una cantidad pequeña de requerimientos, sin embargo, para un volumen grande, esta técnica no es la más adecuada.

Ventajas

- Permite determinar el grado de importancia de cada requerimiento.
- Ayuda a identificar conflictos en los requerimientos.
- Muestra el orden en que deben ser implementados los requerimientos.

Desventajas

- Debe construirse un estándar claro de evaluación, que incluya la participación del cliente.

- **Casos de uso**

Según el autor Sommerville, los casos de uso son una técnica que se basa en escenarios para la obtención de requerimientos. Actualmente, se han convertido en una característica fundamental de la notación UML, que se utiliza para describir modelos de sistemas orientados a objetos.

Los casos de uso son una técnica para especificar el comportamiento de un sistema.

Los casos de uso permiten entonces describir la posible secuencia de interacciones entre el sistema y uno o más actores, en respuesta a un estímulo inicial proveniente de un actor, es una descripción de un conjunto de escenarios, cada uno de ellos comenzado con un evento inicial desde un actor hacia el sistema. La mayoría de los requerimientos funcionales, sino todos, se pueden expresar con casos de uso.

Ventajas

- Representan los requerimientos desde el punto de vista del usuario.
- Permiten representar más de un rol para cada afectado.
- Identifica requerimientos estancados, dentro de un conjunto de requerimientos.

Desventajas

- En sistemas grandes, toma mucho tiempo definir todos los casos de uso.
- El análisis de calidad depende de la calidad con que se haya hecho la descripción inicial.

Entrevistas y Casos de Uso: Un alto porcentaje de la información recolectada durante una entrevista, puede ser usada para construir casos de uso. Mediante esto, el equipo de desarrollo puede entender mejor el ambiente de trabajo de los involucrados. Cuando el analista sienta que tiene dificultades para entender una tarea, pueden recurrir al uso de un cuestionario y mostrar los detalles recabados en un caso de uso. De hecho, durante las entrevistas cualquier usuario puede utilizar diagramas de casos de uso para explicar su entorno de trabajo.

Entrevistas vs. Lluvia de Ideas: Muchas de las ideas planteadas en el grupo, provienen información recopilada en entrevistas o cuestionarios previos. Realmente la lluvia de ideas trata de encontrar las dificultades que existen para la comprensión de términos y conceptos por parte de los participantes; de esta forma se llega a un consenso.

Casos de Uso vs. Lluvia de Ideas: La lista de ideas proveniente del Brainstorm puede ser representada gráficamente mediante casos de uso. **[2]**

	Análisis del Problema	Evaluación y negociación	Especificación de Requisitos	Evaluación	Evolución
Entrevistas y Cuestionarios	x				x
Lluvia de Ideas	x	x			x
Prototipos				x	
Análisis Jerárquico		x			x
Casos de Uso	x		x		x

Figura 4: Técnicas que pueden ser utilizadas en las diferentes actividades de la IR

En esta tabla se observa que la actividad de evolución de los requerimientos aunque depende de las demás es fundamental en la IR debido a que es en esta donde se da mantenimiento al

software de manera general. Al mismo tiempo para el desarrollo de esta actividad se utiliza la mayor cantidad de técnicas que se aplican tanto a los clientes o usuarios finales como a todo aquel que interviene en el desarrollo del producto con el objetivo de obtener un software de mayor calidad y que responda a lo establecido por el cliente.

Las herramientas CASE se le conoce a todo aquel software que es usado para ayudar a las actividades del proceso de desarrollo del software, en donde se ubica la ingeniería de requerimientos. Estas herramientas se concentran en capturar requerimientos, administrarlos y producir una especificación de requisitos.

Es importante hacer ver que estas herramientas fungen como un medio facilitador para agilizar y mejorar los procesos involucrados en todo el ciclo de vida presentado por la IR, y que en conjunto ayudan a la construcción final de un producto de software terminado. **[12]**

Persisten numerosas de estas que corren bajo Sistemas Windows. No existe ninguna aplicación que se encargue de dichas funcionalidades de una manera exacta y precisa. Sin embargo son muy escasas en los sistemas de GNU/Linux, debido a eso los usuarios se ven obligados, al igual que la mayoría de los usuarios de Windows de otro modo a la utilización de una suite ofimática como herramienta de gestión de los requisitos con todos los problemas que eso conlleva.

Estas herramientas permiten entre otras cosas tener un mayor control en proyectos complejos, reducir costos y retrasos en los proyectos, ayudan a determinar la complejidad y los esfuerzos necesarios. Algunas de estas herramientas son:

1.6.1. Open Source Requirement Management Tool (OSRMT)

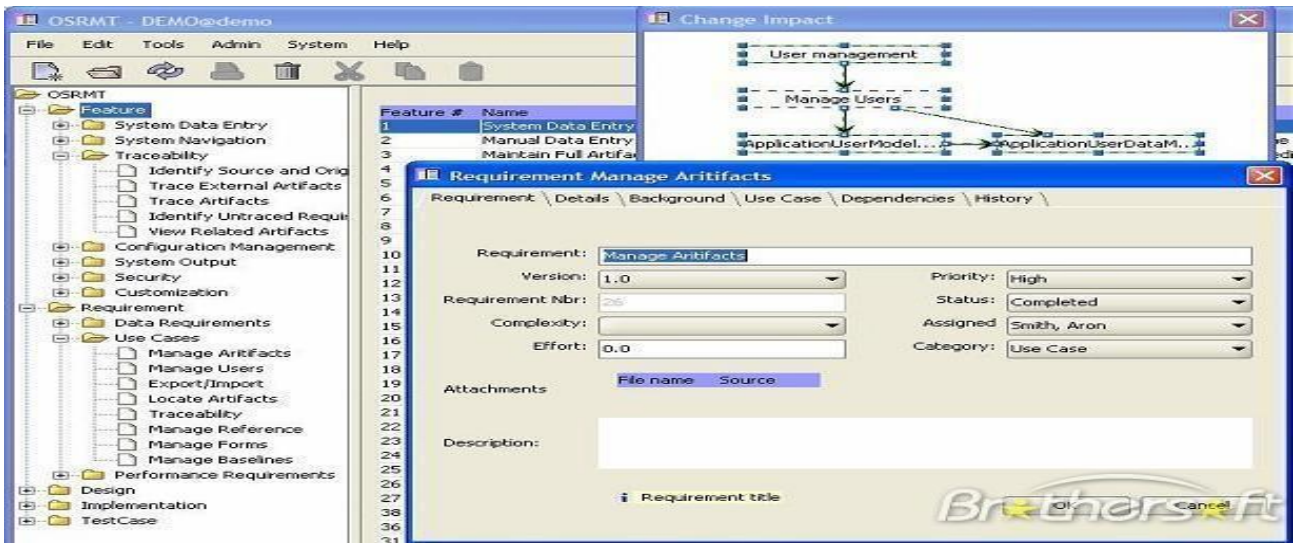


Figura 5: Herramienta OSRMT

Es una herramienta de software libre, bajo licencia GPL, escrita en java y desarrollada actualmente por Aron Smith y Paul Spencer. Se trata de una herramienta de gestión de requisitos, que permite la descripción avanzada de diversos tipos de requisitos y garantiza la trazabilidad entre todos los documentos relacionados con la ingeniería de requisitos (funcionalidades, requisitos, casos de uso, casos de prueba).

Ventajas

- La visualización de requisitos en forma jerárquica es intuitiva y fácil de manejar.
- Existen diversas distribuciones, tanto para un equipo en local como para un servidor de aplicaciones J2EE para permitir desarrollo colaborativo.
- Su licencia es GPL.
- Es un desarrollo basado en Java, por lo que es multiplataforma.
- Las nuevas versiones incorporan un cliente Web para permitir accesos desde internet.
- Como herramienta open source de gestión de requisitos no tiene mucha competencia en cuanto a la funcionalidad ofrecida.
- Tiene una buena documentación.
- El ritmo de mejoras y nuevas versiones es constante.
- Existen muchas opciones para configurar y personalizar la herramienta a las necesidades concretas de una organización.
- Lleva incorporado un sistema de gestión de la configuración que permite definir líneas base.
- Existe un gran soporte para mantener la trazabilidad entre los documentos.
- Existen mecanismos que facilitan la importación y exportación de la información en XML.

Los principales inconvenientes que se han observado son los siguientes:

- No existe un soporte empresarial.
- Las nuevas versiones no están planificadas ni se anuncian claramente las mejoras que serán incorporadas. Es posible que las nuevas versiones no sean compatibles con las anteriores.
- No es posible generar automáticamente un documento de requisitos para entregar al cliente.
- Algunas funcionalidades no han sido desarrolladas completamente y están a medias.
- La interfaz de usuario es en ocasiones lenta.
- Se ofrecen pocos mensajes de confirmación y aviso al usuario (la interacción con el usuario es pobre).

Dentro de las herramientas open source que abarcan la IR, se trata sin duda de una de las que mejores funcionalidades ofrece. Con muchas opciones de configuración, que permite personalizar la herramienta a las necesidades concretas de una organización, esta aplicación cubre los principales aspectos relacionados con la gestión de requisitos: su registro, definición, categorización, seguimiento y trazabilidad con el resto de documentos de trabajo.

Su principal inconveniente radica en que se trata de un proyecto llevado a cabo únicamente por dos desarrolladores, por lo que se hace preciso de un entorno empresarial que permita dar sostenibilidad al proyecto. Su reciente inclusión dentro de la forja sourceforge puede disminuir este riesgo, permitiendo que nuevos desarrolladores colaboren y que el sistema sea sostenible en el tiempo. **[20]**

1.6.2. Requirements Management (REQMAN)

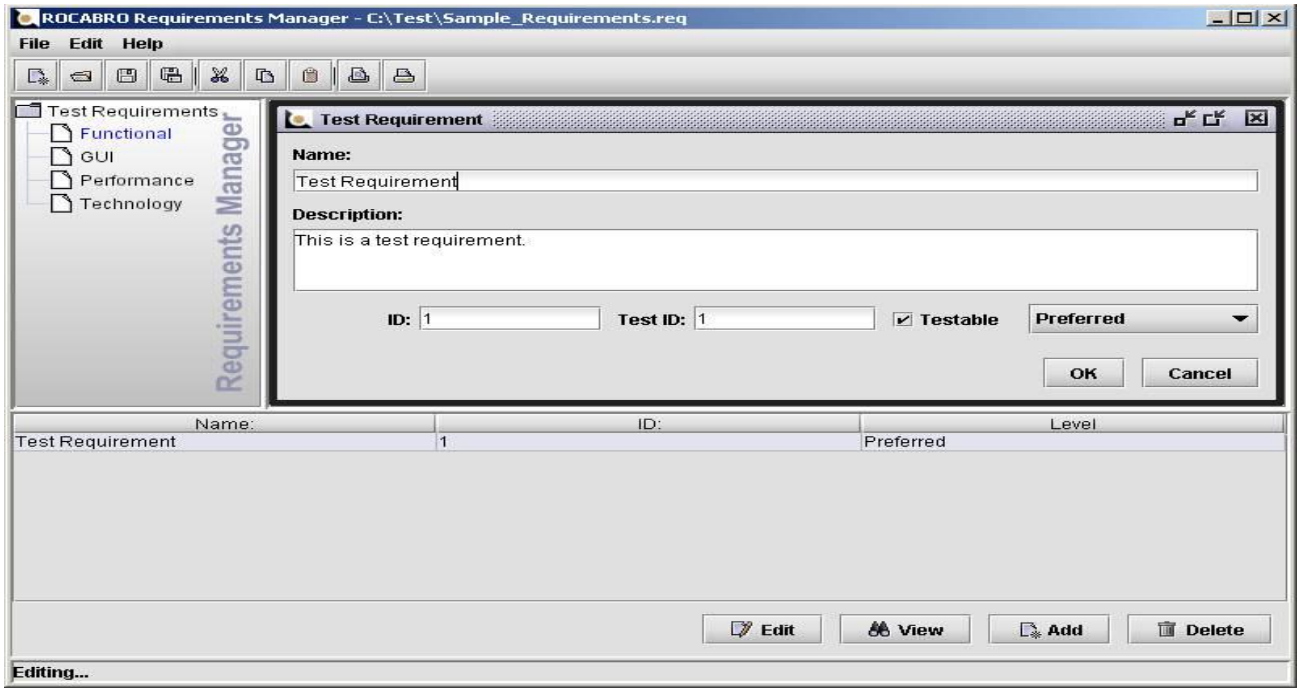


Figura 6: Herramienta ReqMan

Es una solución estructurada donde se guardan todos los datos en un solo lugar central que todo el mundo con el permiso necesario puede acceder desde cualquier lugar y donde todos los cambios tienen su origen hasta el más mínimo detalle.

La plataforma ReqMan es una solución de gestión de proyectos genéricos que se ha adaptado a cada una de estas diferentes aplicaciones, ofrece toda la funcionalidad que necesita para gestionar sus proyectos de principio a fin. Esta es modular, basada en web para el desarrollo de requisitos y la gestión electrónica, los procesos entre las distintas partes interesadas. Consta de

los siguientes módulos que se pueden utilizar de manera independiente o en conjunto:

- Módulo obligatorio
- Módulo del usuario
- Módulo de Información
- Módulo Cuestionario
- Invitación a la Licitación módulo
- Módulo de flujo de trabajo

El módulo más importante es el obligatorio, ya que las salidas de información son vitales para los otros módulos. Antes de utilizar estos, se debe seleccionar el proyecto que está trabajando. Los proyectos se resaltan en rojo en el menú ReqMan.

Una cuenta de cliente puede tener una serie de proyectos y cada uno de estos proyectos puede tener una serie de especificaciones. Cada especificación puede tener una serie de requisitos y de los distintos módulos se utilizan para crear, mantener, visualizar y comunicar los datos de estas especificaciones. Se incluye un glosario al final de este documento. No hay versiones más antiguas. ReqMan realiza un seguimiento de todos los cambios y le da una trazabilidad completa - hasta el más mínimo detalle.

Ventajas

ReqMan no sólo reduce la cantidad de trabajo requerido, sino que también reduce significativamente el riesgo y mejora el resultado final de los proyectos requisito:

- Menor tiempo de comercialización.
- Mayor rendimiento del proyecto.
- Una mejor y más rápida comunicación del equipo.
- Reducción de costos.
- Objetivos definidos.
- Claro, no redundante.
- Reducción del tiempo dedicado al trabajo de proyectos administrativos.
- Mejora de los beneficios generales de los proyectos. **[21]**

1.7. Fases y actividades del PDSW de las distribuciones de GNU/Linux

En la actualidad no existe específicamente un PDSW definido para los sistemas GNU/Linux, sino pasos a tener en cuenta para la construcción de los mismos. Sin embargo no es raro observar que

los distintos PDSW que existen se pueden dividir en tres fases genéricas, con independencia del área de aplicación, tamaño o complejidad del proyecto, por lo que estas se pueden definir como fases del PDSW de las distribuciones de GNU/Linux y son:

1. **Fase de definición:** Es donde los esfuerzos del grupo de trabajo del proyecto se centran en la definición del qué, es decir, se intentan identificar los aspectos de función y rendimiento que deben caracterizar el producto final. El punto clave es la identificación de los requisitos primordiales del sistema a generar.
2. **Fase de desarrollo:** Es donde los esfuerzos del grupo de trabajo del proyecto se centran en el cómo, es decir, se realiza el trabajo de implementación de cada una de las funciones y procedimientos planteados durante la fase de definición.
3. **Fase de mantenimiento:** Es donde los esfuerzos del grupo de trabajo del proyecto se centran en el Cambio. Es decir, en todos los temas relacionados con la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software y a cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente.

1.7.1. Definición

Determinación de las necesidades del cliente

A través de técnicas de recopilación de información se determinan las necesidades del cliente y se almacenan en un listado de requisitos del cliente.

Selección de la distribución anfitrión

El sistema LFS se construirá utilizando una distribución Linux ya instalada (como Debian, Mandriva, RedHat o SUSE, etc.). Este sistema Linux existente (el anfitrión) se utilizará como punto de inicio para suministrar los programas necesarios, como un compilador, un enlazador y un intérprete de comandos, para construir el nuevo sistema. Para esto se tiene en cuenta también lo que necesite el cliente.

Definición de desarrollos propios

Para que un sistema sea único debe tener desarrollado software propio por lo que es en esta fase donde se definen los mismos para desarrollarlos durante la construcción del producto y al igual que en nuestra primera actividad se almacenan las especificaciones que el cliente necesite para estos en un listado de requisitos de desarrollos propios.

1.7.2. Desarrollo

Paquetes y parches

Existen paquetes que se han de descargar para construir un sistema Linux básico. Será necesario

guardar todos los paquetes y parches descargados en algún sitio que esté disponible durante toda la construcción. También se necesita un directorio de trabajo en el que se desempaqueten las fuentes y construirlas.

Aparte de los paquetes, también se necesitan varios parches. Estos parches corrigen pequeños errores en los paquetes que debería solucionar su desarrollador. Los parches también hacen pequeñas modificaciones para facilitar el trabajo con el paquete.

Construir un sistema temporal

La construcción de este sistema minimalista se hará en dos etapas. La primera es construir un conjunto de herramientas independiente del sistema anfitrión (compilador, ensamblador, enlazador, librerías y unas pocas herramientas útiles). La segunda etapa utiliza estas herramientas para construir el resto de herramientas esenciales.

Varios de los paquetes deben parchearse antes de compilarlos, pero sólo cuando el parche es necesario para solucionar un problema. Durante la compilación de muchos paquetes verán aparecer en pantalla diversos avisos (warnings). Esto es normal y se puede ignorar con tranquilidad. No son más que eso, avisos; la mayoría debidos a un uso inapropiado, pero no inválido, de la sintaxis de C o C++. Se debe a que los estándares de C cambian con frecuencia y algunos paquetes todavía usan un estándar antiguo. Esto no es un problema, pero hace que se muestre el aviso.

Selección e instalación de los programas del sistema base

En esta actividad se entra en la zona de edificación y se comienza a construir de verdad el sistema. Es decir, se cambia la raíz del mini sistema Linux temporal, se hace unos cuantos preparativos finales, y entonces se comienza a instalar los paquetes. La clave para aprender qué hace que un sistema Linux funcione es conocer para qué se utiliza cada paquete y por qué el usuario (o el sistema) lo necesita.

Administración de paquetes

Un administrador de paquetes permite supervisar la instalación de ficheros facilitando la eliminación y actualización de ficheros. Un administrador de paquetes facilita la actualización a nuevas versiones cuando estas son liberadas.

Técnicas comunes de administración de paquetes

- **¡Todos está en mi cabeza!**
- **Instalar en directorios separados**
- **Administración de paquetes por medio de enlaces**.
- **Basado en marcas**.
- **Basado en LD_PRELOAD**.

- **Crear archivos de paquetes**
- **Administración basada en usuario**.

Configurar los guiones de arranque del sistema

Muchos de estos guiones funcionarán sin necesidad de modificarlos, pero algunos necesitan ficheros de configuración adicionales, pues manejan información dependiente del hardware. Linux utiliza como sistema de inicio SysVinit, que se basa en el concepto de niveles de ejecución. Este sistema de inicio puede variar ampliamente de un sistema a otro, por lo tanto no se debe asumir que porque las cosas funcionen en una distribución en concreto tengan que funcionar en LFS también.

Hacer el sistema arrancable

Se crea un fichero fstab, la construcción de un núcleo para el nuevo sistema LFS y la instalación del gestor de arranque GRUB para que el sistema LFS se pueda seleccionar para arrancar al inicio. El arranque puede ser una tarea compleja.

Probar sistema instalable

Se realizan las pruebas pertinentes para determinar las deficiencias del sistema dentro de estas están las pruebas de rendimiento, seguridad, pruebas de funcionalidad, en esta última en caso de existir alguna no conformidad queda plasmada en una plantilla de no conformidades. **[22]**

1.7.3. Mantenimiento

Desplegar sistema

El despliegue del software son todas las actividades que se le hacen al sistema de software disponible para el uso. Estas actividades están correlacionadas con posibles transiciones entre ellas.

Luego de haber realizado un análisis completo sobre las tendencias de la IR a nivel internacional, en Cuba, en la UCI y específicamente en Nova, además de estudios de herramientas, técnicas, modelos, metodologías, etc, que contemplan la IR se decide recomendar para la gestión de requisitos en las distribuciones de GNU/Linux la herramienta **OSRMT** ya que es una herramienta que corre bajo Licencia publica general (GPL) y dentro de las herramientas Open Source es sin duda unas de las que más funcionalidades ofrece. Para la especificación de requisitos se recomienda seguir la norma IEEE 610.12 complementada con la IEEE 830 producto a los beneficios de obtener una buena ERS en conjunto con las buenas prácticas del área de administración de requisitos del modelo CMMI (2).

Capítulo 2: Procesos de IR en el desarrollo de distribuciones GNU/Linux

En el capítulo anterior se estudia el PDSW que siguen las distribuciones GNU/Linux con el objetivo de identificar en el presente capítulo las fases del proceso en las que se contempla la IR, específicamente las actividades de cada fase donde se observa. A partir de esto se relacionan los procesos genéricos de IR investigados en el capítulo 1 con el desarrollo de las distribuciones GNU/Linux, se definen los PIR para distribuciones GNU/Linux y se explica cómo ejecutar cada uno de estos en los sistemas GNU/Linux.

2.1. Relación de los PIR con el desarrollo de las distribuciones de GNU/Linux

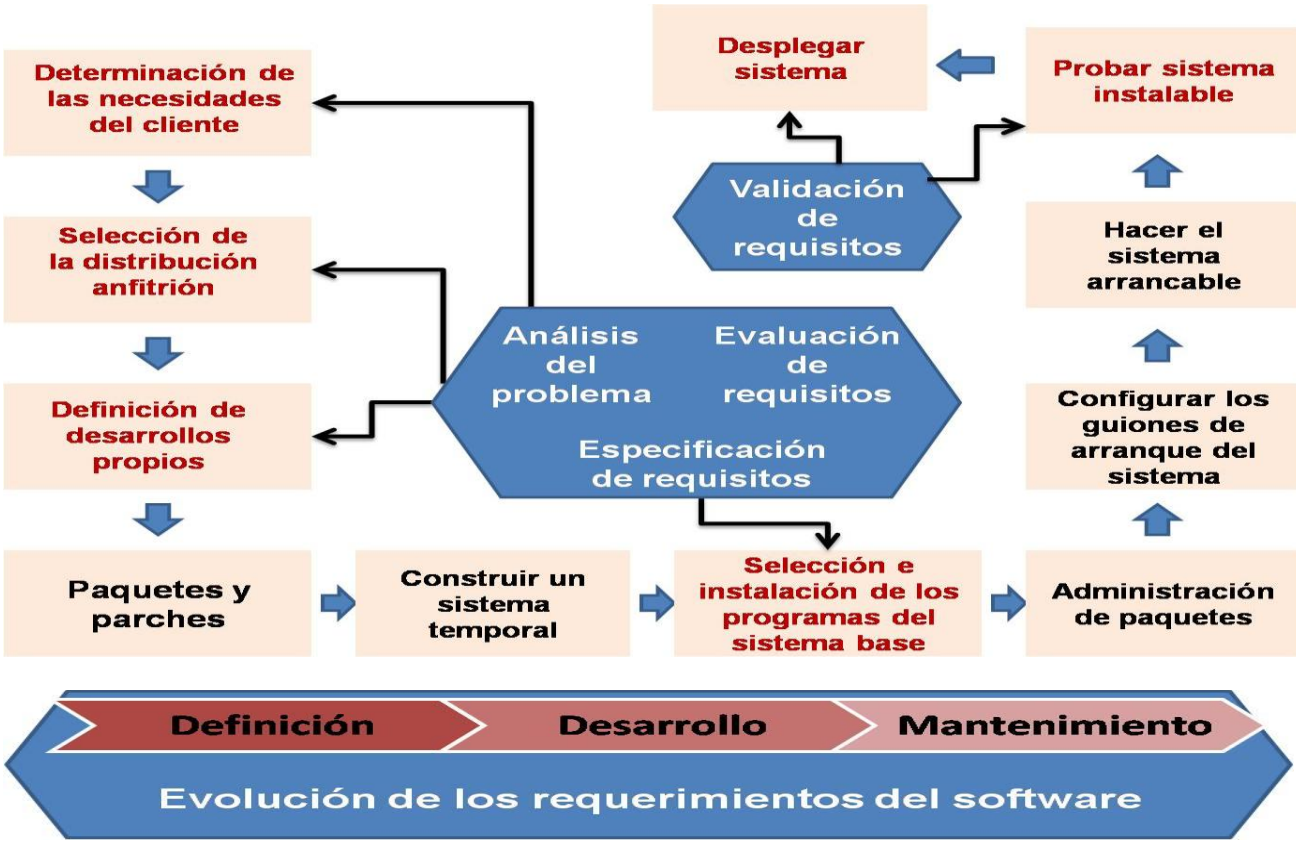


Figura 7: Relación de los PIR con el desarrollo de las distribuciones de GNU/Linux

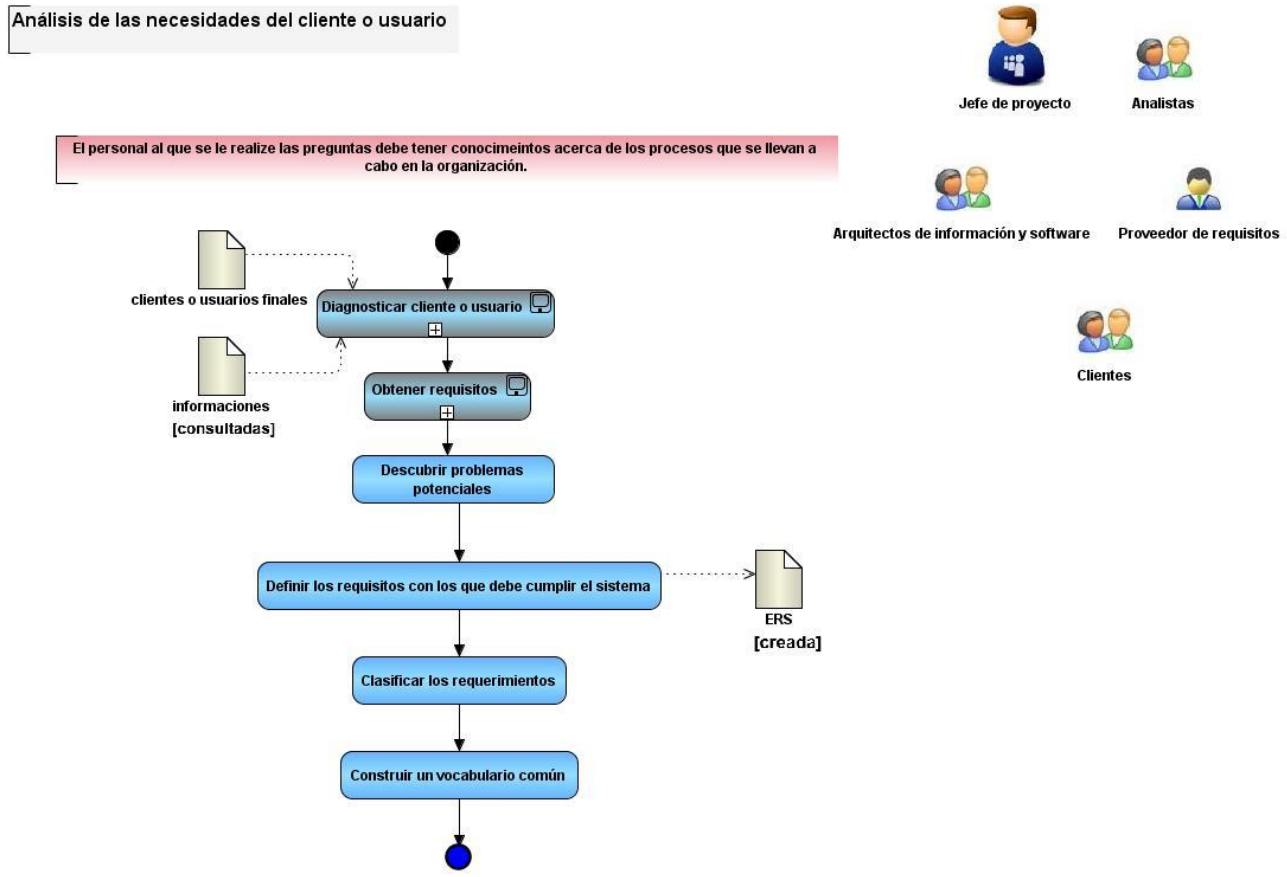
Como se puede observar en la imagen la evolución de los requerimientos es un proceso que se desarrolla durante todo el ciclo de vida del software y depende del transcurso de tiempo, de la aplicación de los métodos definidos y del progreso de los mismos. Las actividades del PDSW que

se muestran con la fuente en roja son las definidas como actividades que contemplan la IR. Las que se muestran en negrita no se relacionan directamente con la IR pero forman parte del PDSW de las distribuciones GNU/Linux. Los procesos análisis del problema, evaluación de requisitos y especificación de requisitos se ejecutan en cada una de las actividades del proceso de desarrollo a las cuales puntean las flechas que se originan en estos procesos, así mismo la evaluación de requisitos.

2.2. Procesos de IR en el desarrollo de distribuciones de GNU/Linux

Para el desarrollo de distribuciones GNU/Linux los PIR definidos son:

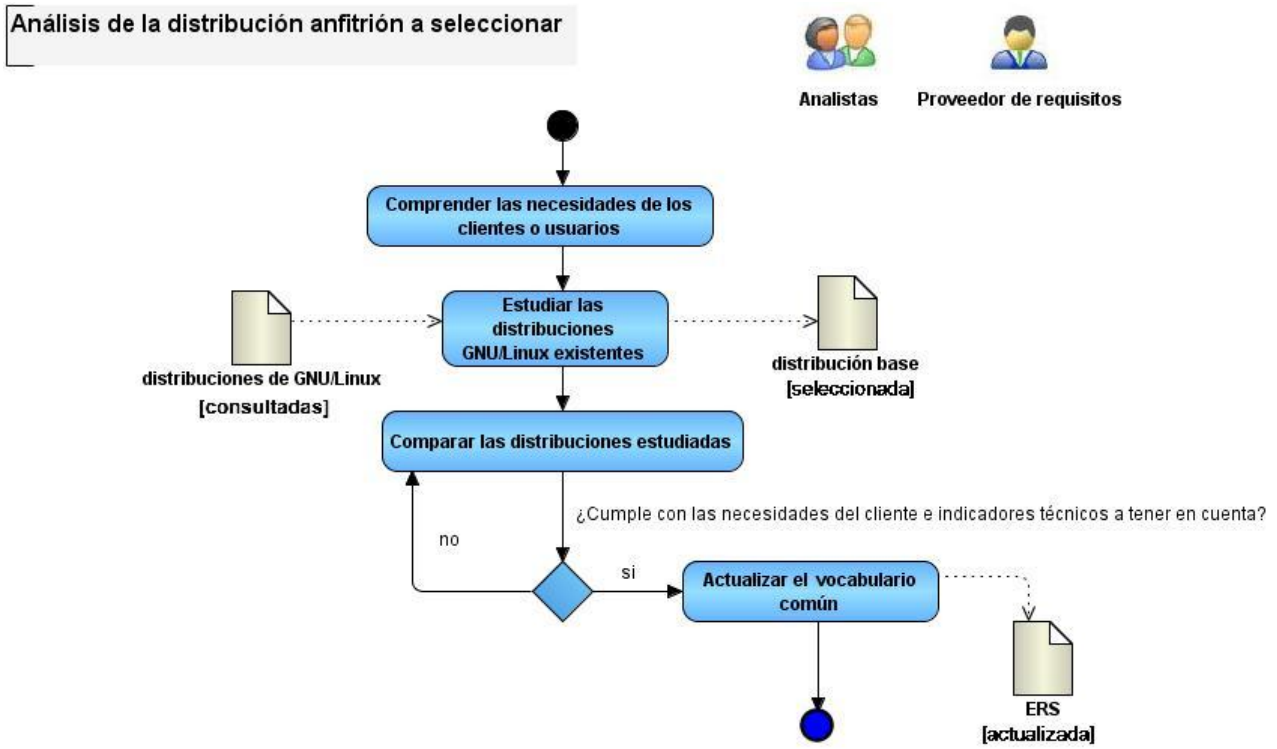
Análisis de las necesidades del cliente o usuario	
Actividades:	**1. Diagnosticar a los clientes o usuarios para estar al tanto sobre sus conocimientos de software libre.** • **Definir las pruebas que se realizan.** **2. Obtener los requisitos que satisfacen las necesidades de los usuarios mediante técnicas de recopilación de información.** • **Definir las preguntas que se le hacen al cliente.** **3. Descubrir problemas potenciales.** **4. Definir los requisitos con los que debe cumplir el sistema.** **5. Clasificar los requerimientos.** **6. Construir un vocabulario común.**
Entradas:	**Clientes o usuarios finales, además informaciones que contenga la organización cliente donde se expliquen los procesos que se llevan a cabo en las mismas.**
Condiciones:	**El personal al que se le realice las preguntas debe tener conocimiento acerca de los procesos que se llevan a cabo en la organización.**
Técnicas:	**Entrevistas y Cuestionarios.** **Lluvia de ideas.**
Salidas:	**ERS.**



Ver (Anexo 1)

Análisis de la distribución anfitrión a seleccionar	
Actividades:	**1. Comprender las necesidades de los clientes o usuarios.** **2. Realizar un estudio de las distribuciones GNU/Linux existentes.** **3. Comparar las distribuciones estudiadas teniendo como indicadores las necesidades del cliente o usuario final y el cumplimiento con determinados factores de calidad.** **4. Actualizar el vocabulario común.**
Entradas:	**Distribuciones de GNU/Linux**
Condiciones:	**Que la distribución seleccionada cumpla con las necesidades del cliente, que sea estable, con la menor cantidad de errores, técnicamente correcta, modificable lo más fácilmente posible.**
Técnicas:	**Entrevistas y Cuestionarios**

Salidas:	**Lluvia de ideas.** **Adición de los requisitos de la nueva distribución a la ERS.** **Distribución base.**



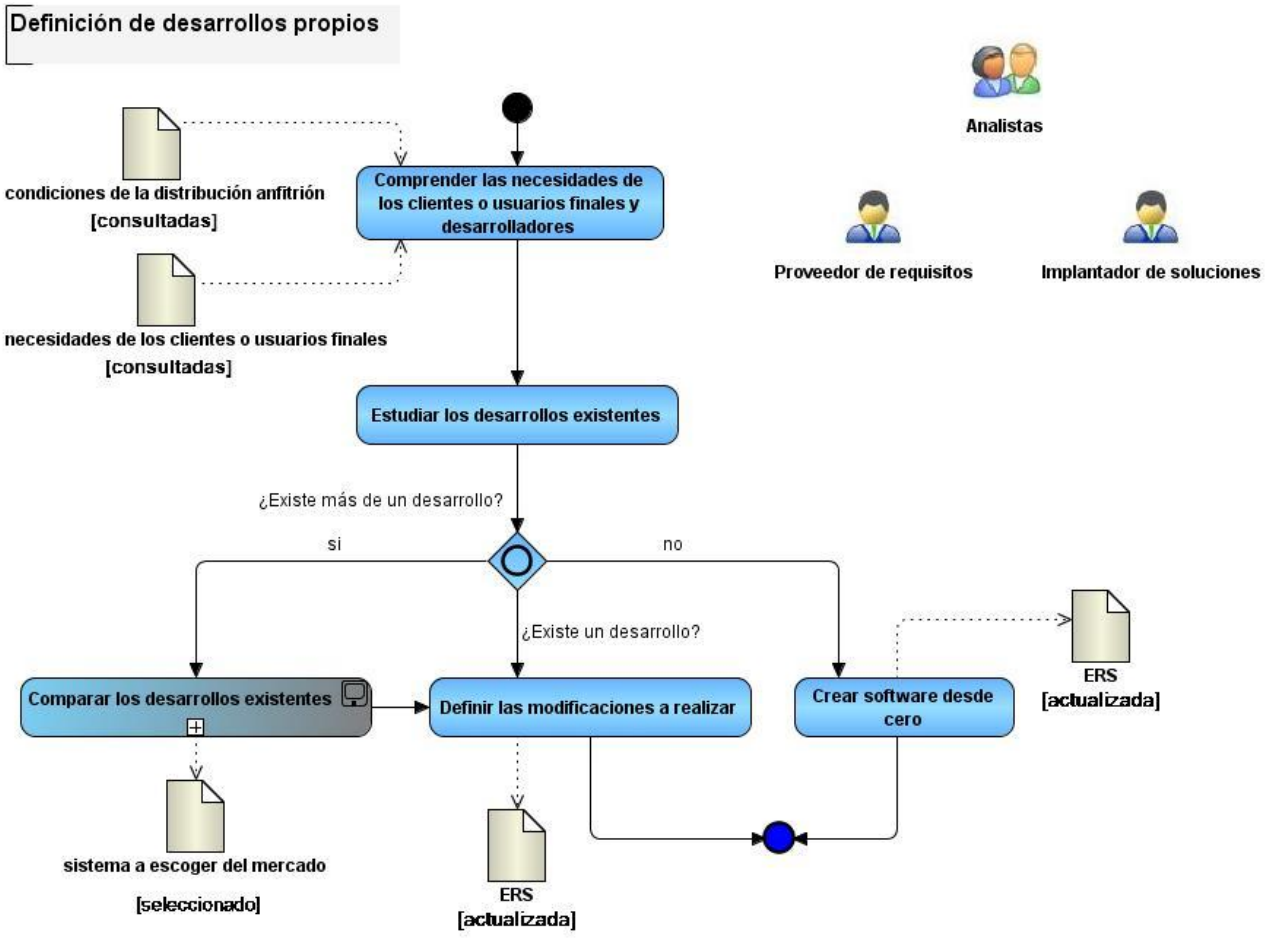
Definición de desarrollos propios

Actividades:

1. **Comprender las necesidades de los clientes o usuarios finales y desarrolladores.**
2. **Estudiar los desarrollos existentes teniendo en cuenta las necesidades del proceso de desarrollo.**
3. **Comparar los desarrollos existentes para determinar el más eficiente.**
   - **Precisar los aspectos a comparar.**
4. **Definir las modificaciones a realizar.**
5. **Crear software desde cero.**
6. **Actualizar el vocabulario común.**

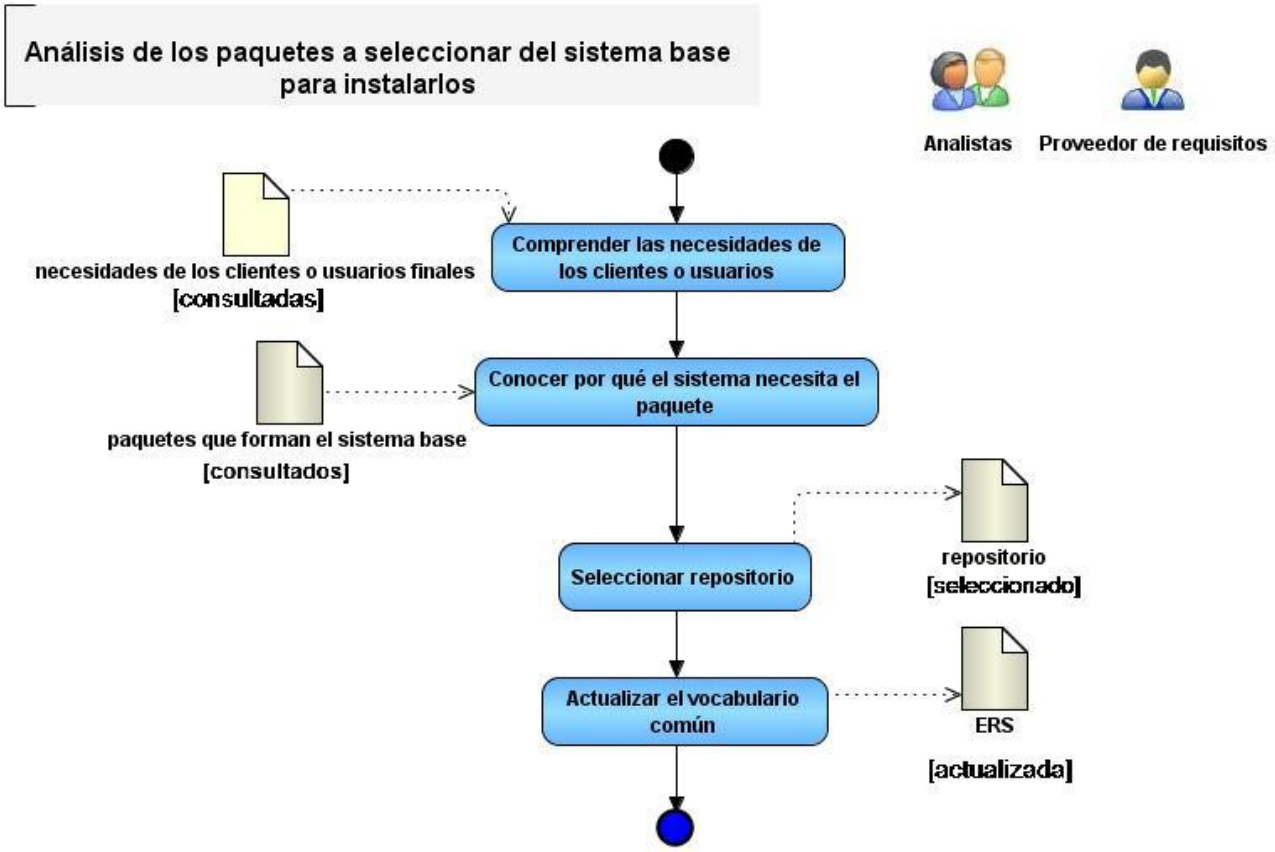
CAPÍTULO 2: PROCESOS DE IR EN EL DESARROLLO DE DISTRIBUCIONES GNU/LINUX

Entradas:	**Condiciones de la distribución anfitrión.** **Necesidades de los clientes o usuarios finales.**
Condiciones:	**Se van a comparar los desarrollos siempre y cuando exista más de uno. En caso de existir solo uno se definen las modificaciones a realizar. Se crea el software desde cero cuando no existan desarrollos para la funcionalidad requerida.**
Técnicas:	**Análisis jerárquico.** **Lluvia de ideas.**
Salidas:	**Adición de los requisitos de cada desarrollo propio a la ERS.** **Sistema a escoger del mercado.**

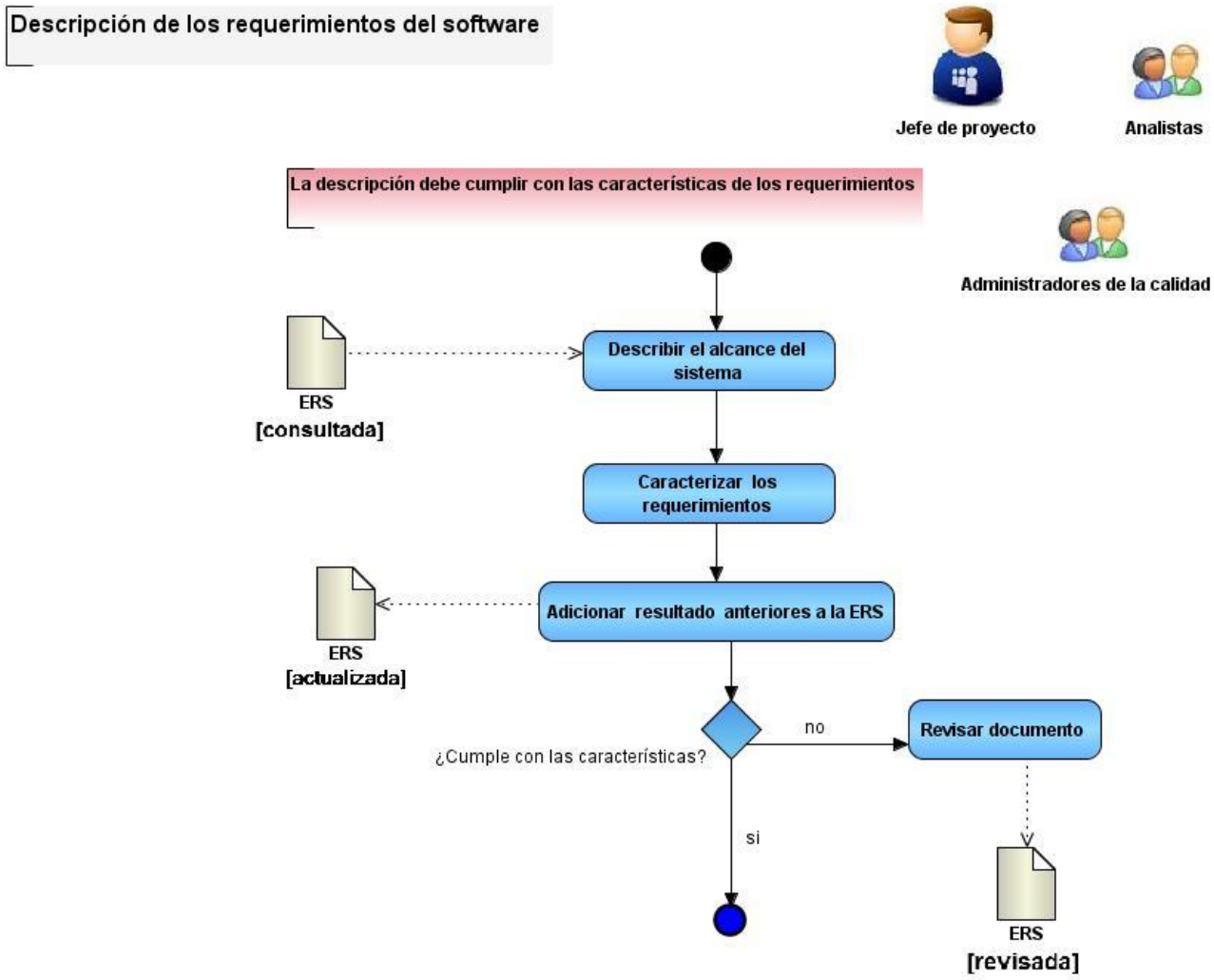


Ver (Anexo 2)

Análisis de los paquetes a seleccionar del sistema base para instalarlos	
Actividades:	1. Comprender las necesidades de los clientes o usuarios finales. 2. Conocer por qué el sistema necesita el paquete. 3. Seleccionar de que repositorio se escogerán los paquetes. 4. Actualizar el vocabulario común.
Entradas:	Paquetes que forman el sistema base. Necesidades de los clientes o usuarios finales.
Condiciones:	
Técnicas:	Entrevistas y cuestionarios. Lluvia de ideas.
Salidas:	Adición de los paquetes necesarios para la distribución a la ERS. Repositorio.

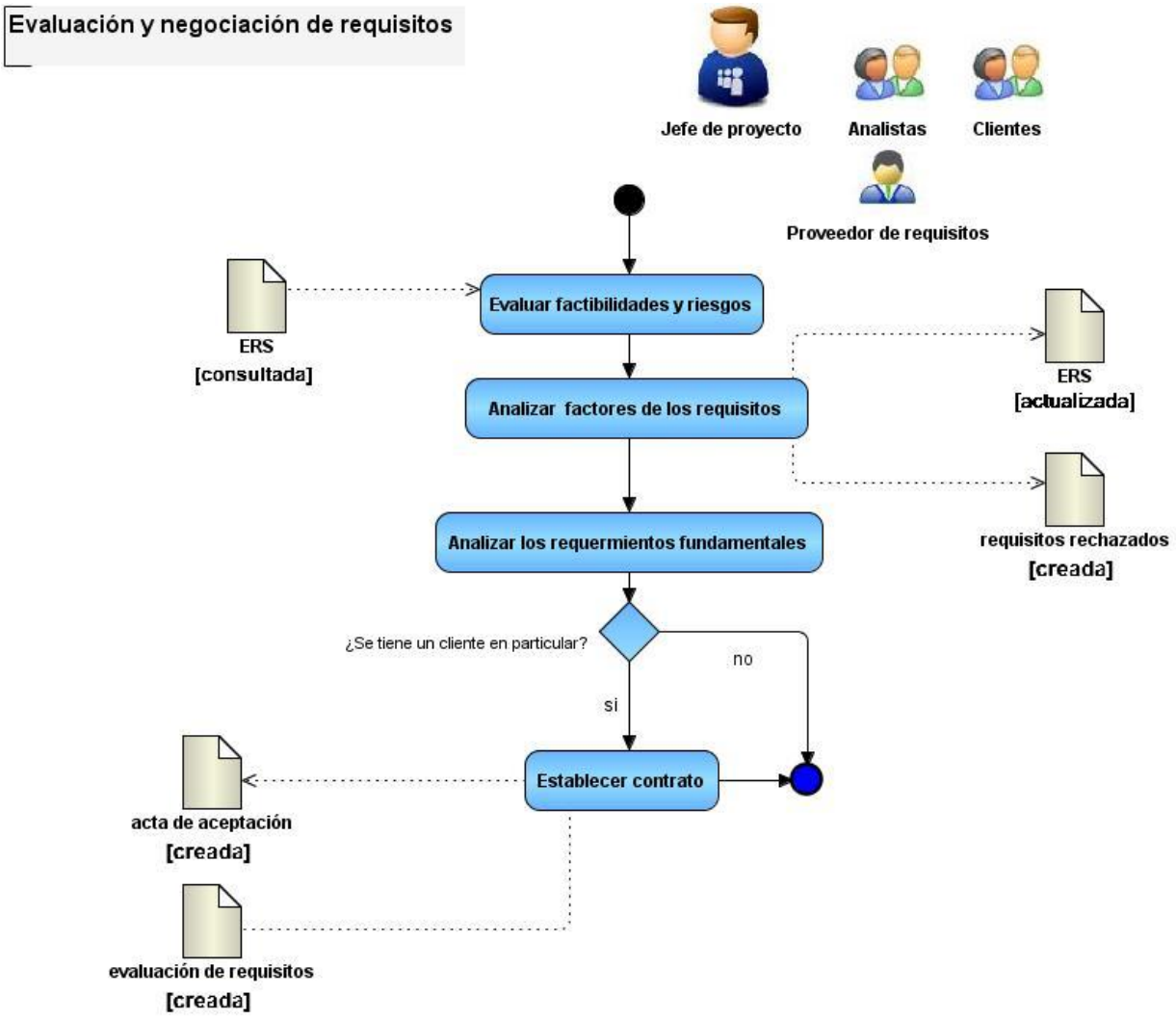


	Descripción de los requerimientos del software
Actividades:	**1. Describir el alcance del sistema.** **2. Definir todos los requerimientos funcionales y no funcionales y cualquier otra información que sirva de soporte y guía para fases posteriores.** **3. Adicionar a la ERS el resultado de las actividades anteriores.** **4. Revisar documento para que logre cumplir con las características que debe tener el mismo.**
Entradas:	**ERS.**
Condiciones:	**Que la descripción cumpla con las características de los requerimientos, para asegurar esto se hace una revisión profunda de la ERS.**
Técnicas:	**Casos de Uso.**
Salidas:	**ERS revisada.**

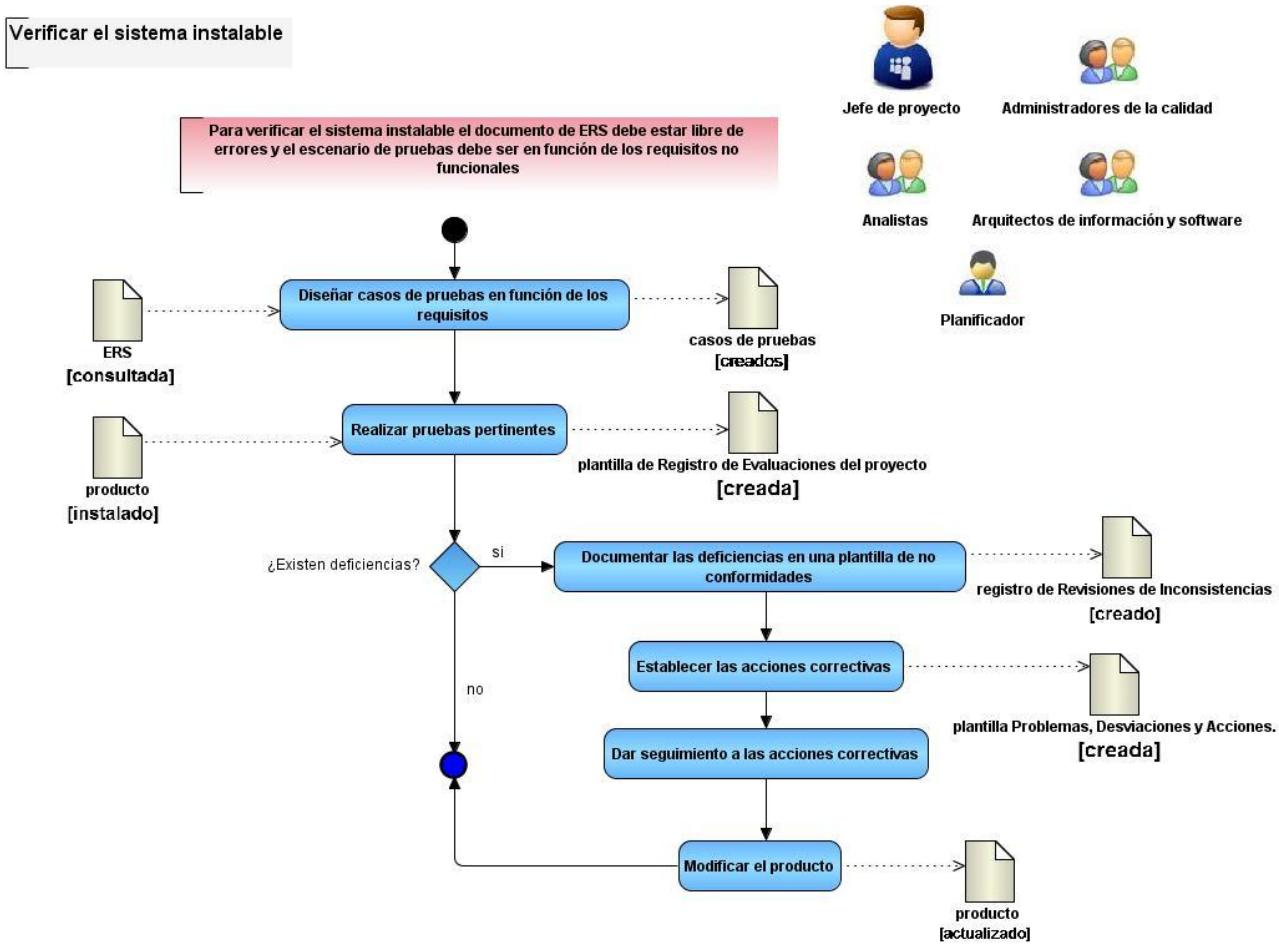


Evaluación y negociación de requisitos	
Actividades:	**1. Evaluar factibilidades y riesgos.** **2. Analizar costo, complejidad, prioridad para ser satisfechos y otros factores de los requisitos para determinar su inclusión.** **3. Analizar con los involucrados relevantes los requerimientos fundamentales para llevar a buen término la solución del problema a plantear.**

	4. Establecer el contrato formal con el cliente.
Entradas:	**ERS.**
Condiciones:	**Si existe un cliente en particular se establece el contrato formal con éste.**
Técnicas:	**Lluvia de Ideas.**
Salidas:	**ERS actualizada.** **Evaluación de requisitos.** **Requisitos rechazados.** **Acta de aceptación.**

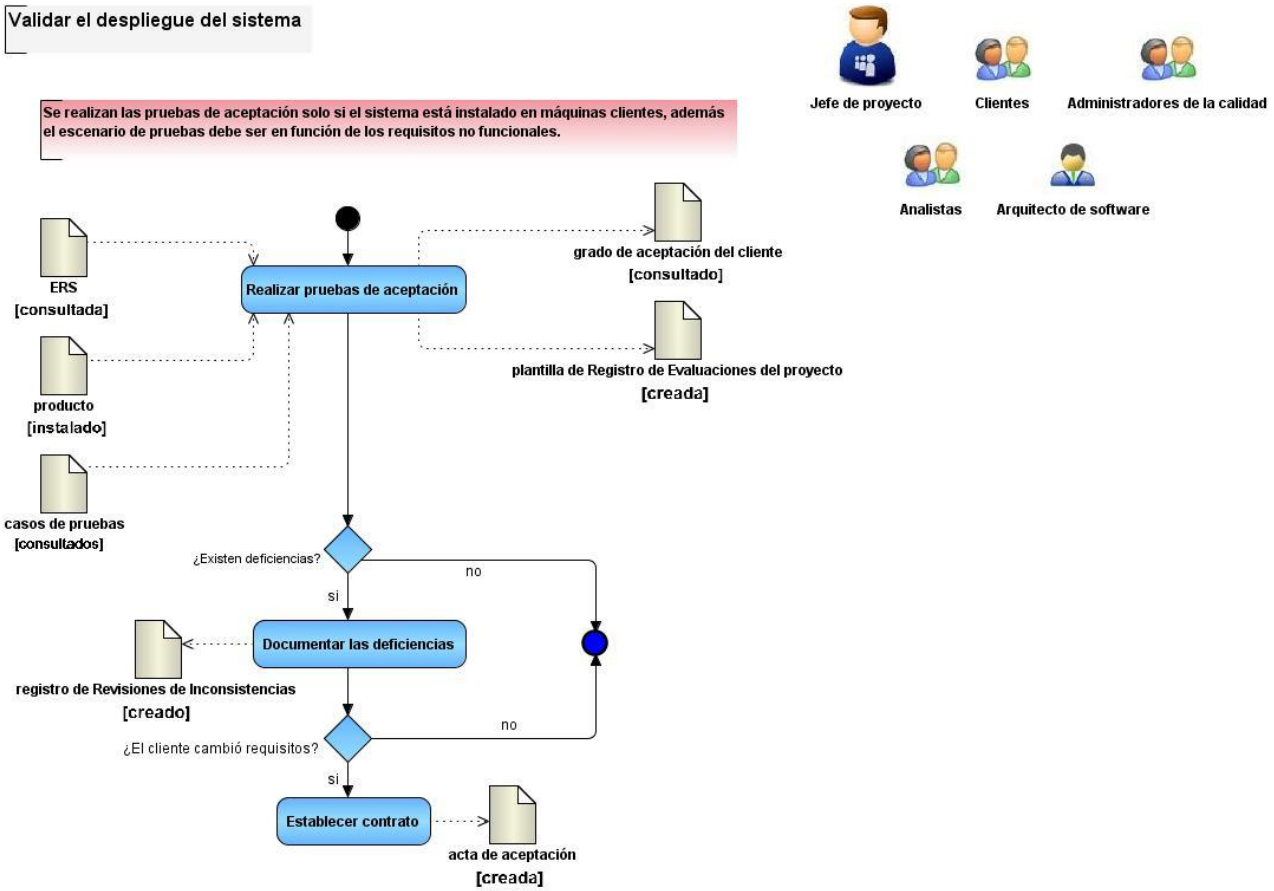


Verificar el sistema instalable	
Actividades:	**1. En base a cada una de las características que se deseen revisar determinar los criterios de evaluación.** **• Elaborar las preguntas a realizar o proponer técnicas que permitan evaluarlas.** **2. Realizar las pruebas pertinentes para determinar las deficiencias del sistema.** **3. Documentar las deficiencias en una plantilla de no conformidades.** **4. Establecer las acciones correctivas.** **5. Dar seguimiento a las acciones correctivas.** **6. Modificar el producto.**
Entradas:	**ERS.** **Producto.**
Condiciones:	**ERS debe estar libre de errores, porque así, la verificación garantiza que todos los requerimientos presentes en el documento de especificación puedan ser verificados correctamente.** **Se crea una plantilla de no conformidades, se establecen acciones correctivas, se da seguimiento a esas acciones y se modifica el producto en caso de existir deficiencias.** **El escenario de pruebas debe ser en función de los requisitos no funcionales.**
Técnicas:	**Revisiones técnicas.**
Salidas:	**Registro de Revisiones de Inconsistencias (RRI).** **Plantilla Problemas, Desviaciones y Acciones.** **Producto modificado.** **Plantilla de Registro de Evaluaciones del proyecto.** **Casos de pruebas.**



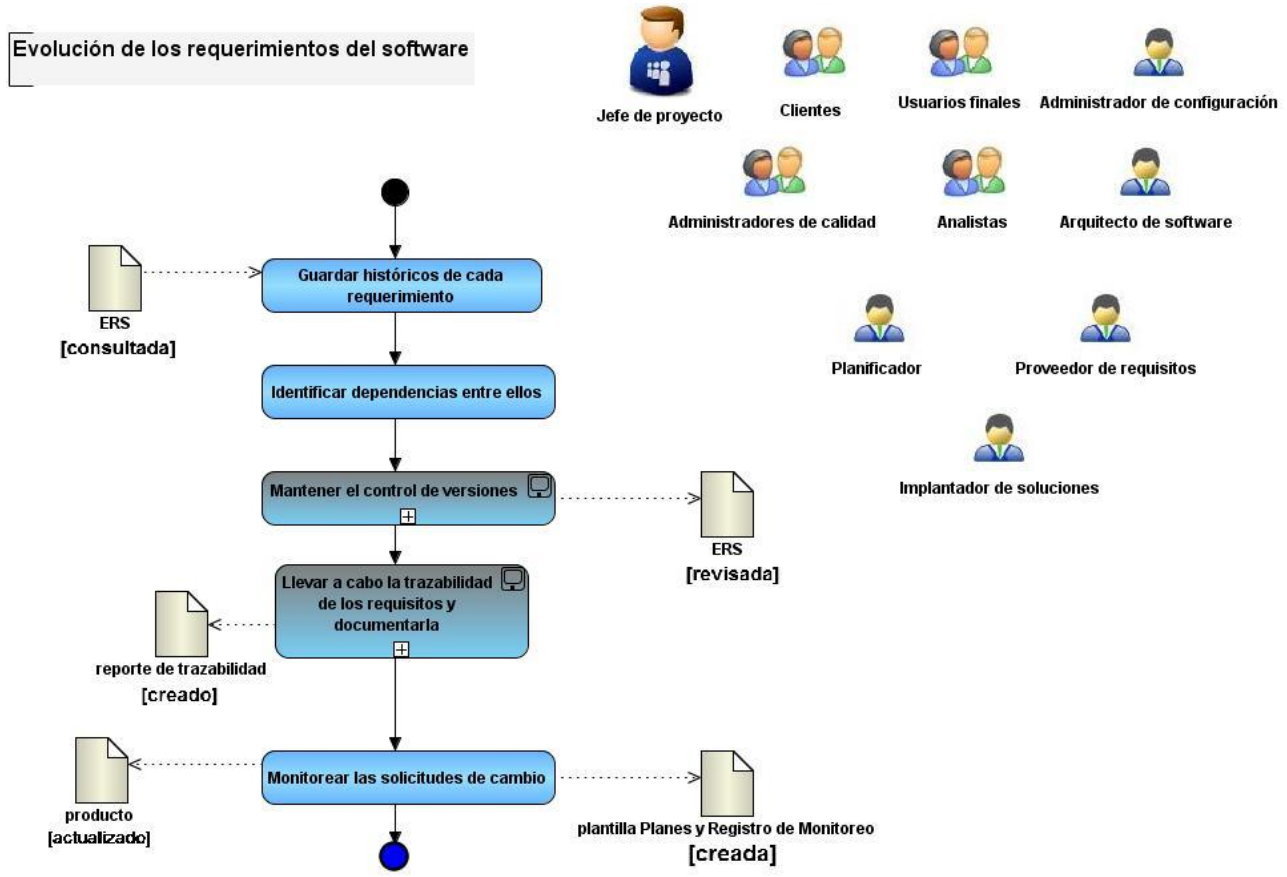
Validar el despliegue del sistema	
Actividades:	**1. Realizar pruebas de aceptación con o como representante de los clientes en los casos necesarios.** **2. Documentar las deficiencias en una plantilla de no conformidades.** **3. Establecer otro contrato con nuevos requisitos para modificar el sistema en una nueva versión del software.**
Entradas:	**ERS.** **Producto.** **Casos de pruebas.**
Condiciones:	**Sistema instalado en máquinas clientes.** **El escenario de pruebas debe ser en función de los requisitos no funcionales.**

	Se establece otro contrato en caso de que el cliente cambie algunos requisitos. En caso de existir deficiencias se documentan en una plantilla de no conformidades.
Técnicas:	**Pruebas de software o revisiones técnicas formales.**
Salidas:	**Acta de aceptación.** **RRI.** **Plantilla de Registro de Evaluaciones del proyecto.** **Contrato.**



Evolución de los requerimientos del software

Actividades:	**1. Guardar históricos de cada requerimiento.** **2. Identificar dependencias entre ellos.** **3. Mantener el control de versiones.** • **Poseer las actualizaciones que se le realicen al producto.** • **Realizar pruebas en un período de tiempo establecido para así tener conocimiento de la evolución de los requisitos.** **4. Llevar a cabo la Trazabilidad de los requisitos y documentarla.** • **Identificar los elementos a tracear.** • **Definir la estructura de trazabilidad entre los elementos.** • **Definir las dependencias bidireccionales.** • **Generar la matriz de trazabilidad.** **5. Monitorear las solicitudes de cambio.**
Entradas:	**ERS.**
Condiciones:	
Técnicas:	**Entrevistas y cuestionarios.** **Lluvia de Ideas.** **Revisiones.**
Salidas:	**ERS revisada.** **Reporte de Trazabilidad.** **Plantilla Planes y Registro de Monitoreo.** **Producto modificado.**


Evolución de los requerimientos del software
Jefe de proyecto
Clientes
Usuarios finales
Administrador de configuración
Administradores de calidad
Analistas
Arquitecto de software
Planificador
Proveedor de requisitos
Implantador de soluciones
Guardar históricos de cada requerimiento
ERS
[consultada]
Identificar dependencias entre ellos
Mantener el control de versiones
ERS
[revisada]
Llevar a cabo la trazabilidad de los requisitos y documentarla
reporte de trazabilidad
[creado]
Monitorear las solicitudes de cambio
producto
[actualizado]
plantilla Planes y Registro de Monitoreo
[creada]

Ver (Anexo 3)

2.3. ¿Cómo ocurren los procesos de IR en el desarrollo de distribuciones de GNU/Linux?

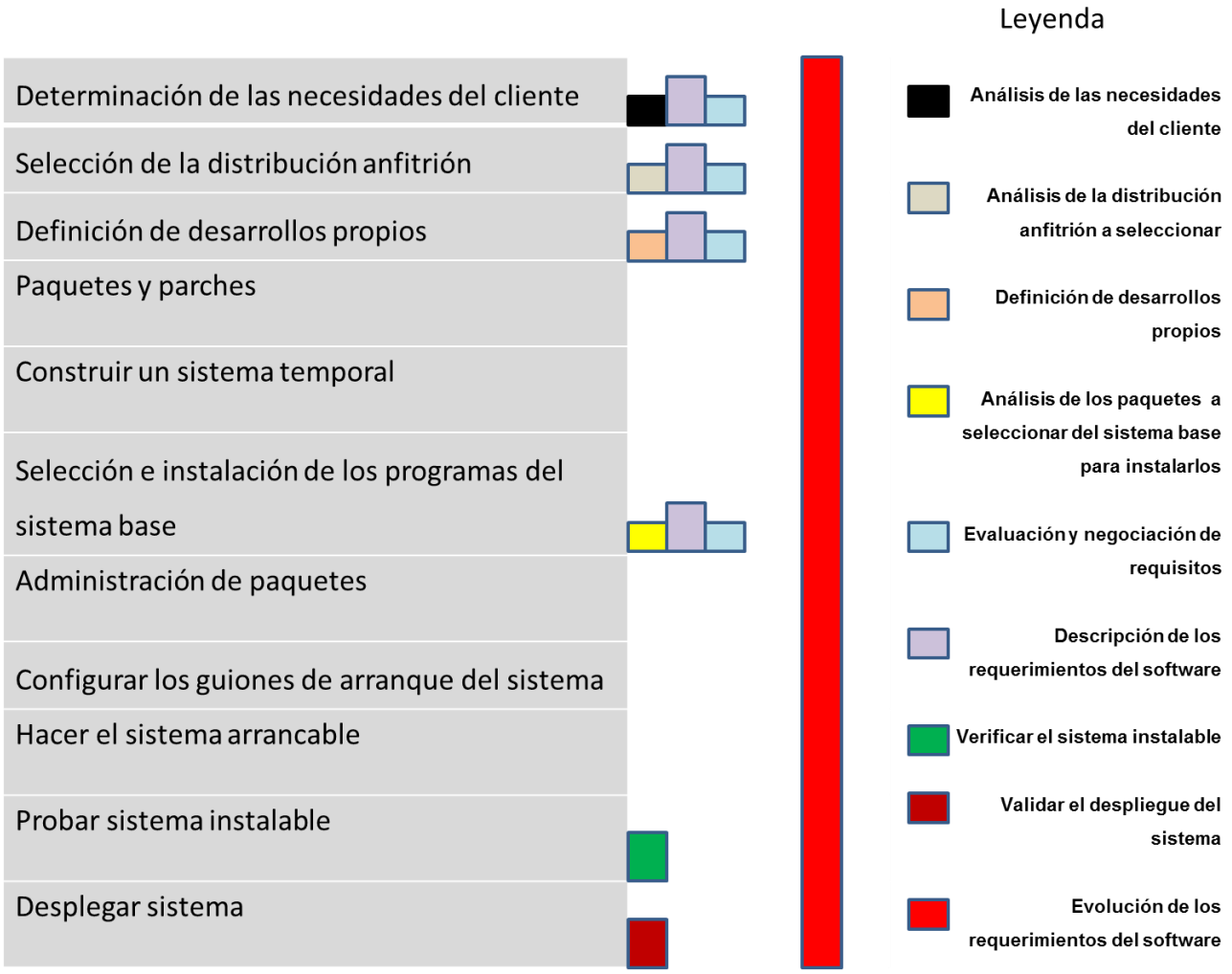


Figura 7: Relación de los procesos de IR

Como se puede apreciar en la gráfica los procesos análisis de las necesidades del cliente, análisis de la distribución anfitrión a seleccionar, definición de desarrollos propios y análisis de los paquetes a seleccionar del sistema base para instalarlos se ejecutan al mismo tiempo que la descripción de los requerimientos del software y la evaluación y negociación de requisitos. La descripción de los requisitos culmina su período de desarrollo luego de terminadas las anteriores producto a las transformaciones que ofrece la evaluación y negociación a la especificación además de estar en constante modificación por los resultados de las actividades de análisis. La

respuesta del por qué los procesos anteriores se desarrollan a la vez es sencillamente por las salidas que estos generan. Luego de culminado estos, se lleva a cabo la verificación del sistema instalable. Es importante aclarar que para que exista una exitosa verificación del sistema instalable y evaluación del despliegue del sistema se debe presentar un buen documento de especificación por lo que estas actividades dependen de la descripción de los requerimientos, estos dos procesos por su parte se ejecutan consecutivamente, ya que primeramente se verifica y luego se evalúa el sistema. En el caso de la evolución de los requerimientos del software tiene su origen en el comienzo del PDSW y se desarrolla a lo largo de todo el ciclo de vida de este. Cada uno de estos procesos se establece en sus actividades correspondientes, son adaptables y genéricos para este tipo de producto, por lo que pueden ser puestos en práctica independientemente de la metodología o modelo que se utilice.

Capítulo 3: Evaluación de los PIR propuestos

Luego de haber diseñado los PIR que pueden ser puestos en práctica durante el desarrollo de distribuciones de GNU/Linux, se evalúan los procesos propuestos, objetivo principal del presente capítulo. Se escoge la distribución cubana GNU/Linux Nova para aplicar los procesos escogidos de la propuesta y evaluar su comportamiento en la misma, por tal motivo en el primer punto de este capítulo se muestra una imagen la cual brinda información del PDSW de esta distribución. Luego se identifica en que actividades se aplican estos procesos para ver si coincide con las actividades del PDSW de las distribuciones de GNU/Linux donde se contempla la IR. Posteriormente se aplica Delphy como método de evaluación por el criterio de especialistas para evaluar la solución que permite ver si la propuesta es correcta de acuerdo al criterio de varios especialistas en el tema, y de personas vinculadas al desarrollo de la distribución. Posteriormente se aplica el método experimental para obtener resultados en la práctica que muestren el impacto de uno de los procesos propuestos.

3.1. Proceso de desarrollo de software de la distribución de GNU/Linux Nova.

El PDSW de la distribución GNU/Linux Nova está compuesto por varios macro procesos que a su vez se componen de un conjunto de actividades que son ejecutadas por determinados roles y para esto reciben una serie de entradas y producen una serie de salidas. Los macro procesos gestión de proyecto, gestión de ambiente, gestión de la configuración de software y gestión de la calidad agrupan las llamadas actividades sombrillas que sirven de apoyo al resto de los macro procesos durante todo el desarrollo del software. La imagen del proceso está disponible en el (Anexo 4)

Las actividades donde se aplican los procesos diseñados son:

- **Levantamiento de requisitos.**
- **Pruebas al sistema.**

3.2. Delphy como método de evaluación por el criterio de especialista.

El método Delphy procede por medio de la interrogación a especialistas con la ayuda de cuestionarios sucesivos, a fin de poner de manifiesto convergencias de opiniones y deducir eventuales consensos.

Las principales características del método son las siguientes:

✓ **Anonimato:** se expresa a través del no-conocimiento de las respuestas, puesto que los miembros del grupo contestan las preguntas sin confrontarse, incluso sin conocerse entre sí.

✓ **Retroalimentación controlada:** después de cada ronda de preguntas se tabulan las respuestas y se procesan de forma tal, que antes de la siguiente ronda los participantes pueden evaluar los resultados de la ronda anterior, así como las razones dadas para cada respuesta y su dispersión del promedio.

✓ **Respuesta estadística del grupo:** entre cada ronda de preguntas, la información obtenida se procesa por medio de técnicas estadístico-matemáticas, como, por ejemplo, el diseño experimental no paramétrico, las que dotan al investigador de un instrumento objetivo y concreto en el cual pueden apoyarse para tomar una decisión final.

3.2.1. Formulación del problema.

Terminada la propuesta de los PIR para el desarrollo de las distribuciones de GNU/Linux, se hace necesario evaluar la correctitud y valor de los mismos, utilizando para ello el método Delphy. Primeramente se definen los atributos (C) a evaluar por los especialistas, estos deben ser precisos, medibles e independientes. Los atributos identificados son los siguientes:

C1. Importancia de la propuesta.

C2. Contribución al mejoramiento del desarrollo y del producto de la distribución seleccionada para evaluar los procesos propuestos.

C3. Contenido de la propuesta.

C4. Aplicación exitosa de los PIR diseñados en la distribución GNU/Linux Nova.

C5. Evaluación de la propuesta.

Los criterios anteriores constituyen las bases para la elaboración del cuestionario presentado a los especialistas.

Siempre se comenzará este proceso enviando un modelo a los posibles especialistas con una explicación breve sobre los indicadores del trabajo y los resultados que se desean obtener. Para la aplicación práctica del método es necesario considerar metodológicamente dos cuestiones fundamentales:

- La selección de los especialistas a encuestar.
- La elaboración del cuestionario.

A continuación se explica cómo fueron aplicadas estas cuestiones.

3.2.2. Selección de los especialistas a encuestar.

Luego de haber decidido aplicar el método Delphy se procede a seleccionar los especialistas en el tema a evaluar. Estos serán personas u organizaciones capaces de ofrecer valoraciones

conclusivas de un problema en cuestión y hacer recomendaciones respecto a sus momentos fundamentales con un máximo de competencia.

Para la selección de los especialistas que determinarán hasta qué punto son correctos los PIR diseñados para las distribuciones de GNU/Linux según sus conocimientos, se tuvieron en cuenta una serie de características que posibilitaron obtener información curricular y actualizada de ellos. Basándose principalmente, en las participaciones de estos en eventos científicos, que contaran además con una ardua experiencia laboral así como sus competencias en el tema. También se tuvo en cuenta los conocimientos que deben tener sobre el problema planteado. Estos conocimientos están basados principalmente en las siguientes áreas de trabajo:

- Proceso de Desarrollo del Software.

Ingeniería de Requisitos.

Listado de especialistas candidatos teniendo en cuenta las anteriores características:

Especialista #1: Jorge Luis Machín Castillo.

Especialista #2: Yurenia Hernández Blanco.

Especialista #3: Yunier Soler Franco.

Especialista #4: Miguel Albalat Águila.

Especialista #5: Daniel Hernández Bahr.

Especialista #6: Sonia Guerrero Lambert.

Especialista #7: Nadia Porro Lugo.

Todos estos especialistas estuvieron de acuerdo en participar en la evaluación de la solución propuesta.

Para que estos especialistas participaran en la evaluación de la propuesta se tuvo en cuenta el nivel de competencia de cada uno de ellos determinado por el coeficiente K, que se calcula haciendo uso de la siguiente fórmula matemática:

K= ½ (kc + ka)

Donde:

K: Coeficiente de competencia.

Kc: es el coeficiente de conocimiento.

Ka: es el coeficiente de argumentación.

El Kc se calcula sobre la valoración del propio especialista en una escala del 0 al 10 y multiplicado por 0,1; de esta forma, la evaluación "0" indica que el especialista que no tiene absolutamente ningún conocimiento de la problemática correspondiente, mientras que la evaluación "10" significa que el especialista tiene pleno conocimiento de la problemática tratada. Entré estas dos

evaluaciones extremas hay nueve intermedias. El especialista deberá marcar con una cruz en la casilla que estime pertinente, por ejemplo:

0	1	2	3	4	5	6	7	8	9	10
									x	

Kc = criterio * 0.1

Grado de conocimiento o información del tema tratado

Especialistas	0	1	2	3	4	5	6	7	8	9	10
1							x				
2									x		
3						x					
4										x	
5							x				
6								x			
7						x					

El resultado de los kc de todos los especialistas se muestra en la siguiente tabla.

Especialistas	1	2	3	4	5	6	7
Kc	0.6	0.8	0.5	0.9	0.6	0.7	0.5

Luego se continúa calculando Ka, el candidato debe clasificar en alto, medio o bajo su grado de competencia sobre los aspectos o fuentes de argumentación sometidos a su consideración. Cada nivel de clasificación posee un valor y la suma de los valores marcados por cada criterio será el coeficiente de argumentación del candidato a especialista, a partir de la tabla patrón definido en un informe titulado Criterio de especialistas: Método Delphy extraído del sitio:

http://eva.uci.cu/file.php/69/Bibliografia_Especilizada_Tema_4/Metodo_DELPHY.pdf

Ka = Σ valoresTP Ver (Anexo 5)

ValoresTP: son los valores de la tabla patrón que se corresponden con lo marcado por los especialistas.

Los resultados de calcular ***Ka*** se muestran en la siguiente tabla.

Especialistas	1	2	3	4	5	6	7
Ka	1	1	0.9	1	0.8	0.9	0.5

Después de calcular los valores de Kc y Ka para cada uno de los especialistas, se procede al cálculo del coeficiente de competencia, a través de la fórmula definida anteriormente. Es recomendable incluir en el grupo a los de coeficiente de competencia alto y medio.

El cálculo **de K** arroja los valores mostrados en la siguiente tabla.

Especialistas	Coeficiente de conocimiento Kc	Coeficiente de argumentación Ka	Coeficiente de competencia K	Grado de influencia del coeficiente de competencia.
1	0.6	1	0.8	Alto
2	0.8	1	0.9	Alto
3	0.5	0.9	0.7	Medio
4	0.9	1	0.95	Alto
5	0.6	0.8	0.7	Medio
6	0.7	0.9	0.8	Alto
7	0.5	0.5	0.5	Medio

El coeficiente de competencia de cada especialista se determina como sigue:

- Si 0,8 ≤ k < 1,0 coeficiente de competencia alto.
- Si 0,5 ≤ k < 0,8 coeficiente de competencia medio.
- Si k < 0,5 coeficiente de competencia bajo.

De los encuestados 4 tienen un coeficiente de competencia alto, 3 medio y ninguno coeficiente bajo. Luego de tener el conjunto de especialistas conformado se invita a cada uno a formar parte de la evaluación de la propuesta y de esta forma conformar el panel de especialistas que serán sometidos al cuestionario elaborado para la evaluación.

3.2.3. Elaboración del cuestionario.

Se hace entrega de la propuesta que se desea evaluar a todos los especialistas para que se documenten sobre el tema de la investigación y luego expresen sus criterios mediante el cuestionario confeccionado.

En el cuestionario elaborado primeramente se solicitan los datos personales a los especialistas, luego se confeccionan 5 preguntas de tipo contable y abierta que contribuyen a la obtención de evaluaciones y criterios de los especialistas; y por último contiene un espacio donde los especialistas pueden emitir su opinión personal sobre los procesos de ingeniería de requisitos propuestos. El cuestionario está disponible en el (Anexo 6).

Las preguntas de tipo contables permiten registrar ciertos aspectos con el objetivo de mostrar cuantitativamente los resultados obtenidos en el cuestionario. En este tipo de preguntas la escala para su respuesta está dividida en 5, sin importar si esta división está propuesta en forma cuantitativa (%) o cualitativa (_Muy Alta, _Alta). El 1 representa el nivel más bajo y el 5 el nivel máximo.

3.2.4. Análisis de los resultados.

Para llevar a cabo este proceso de definen primero un conjunto de criterios de evaluación para hacer un balance de equivalencias con el fin de hacer un análisis general que ofrezca un resultado integrador. Estos criterios son:

Criterios Cualitativos	Criterio expresado en %	Criterio Numérico
Muy alta o Contribuye muy notablemente	100%	5
Alta o Contribuye notablemente	75%	4
Media o Contribuye medianamente	50%	3
Baja o No contribuye muy notablemente	25%	2
Muy baja o No contribuye	0%	1

A continuación se muestran los resultados obtenidos utilizando el método de evaluación por el criterio de especialistas.

Criterios	IP	CM	CP	AE	EP	Total
Especialista 1	**5**	**4**	**4**	**4**	**4**	**21**
Especialista 2	**5**	**4**	**4**	**5**	**5**	**23**
Especialista 3	**5**	**4**	**4**	**5**	**5**	**23**
Especialista 4	**5**	**5**	**4**	**5**	**4**	**23**
Especialista 5	**5**	**4**	**4**	**5**	**4**	**22**
Especialista 6	**5**	**3**	**4**	**4**	**4**	**20**
Especialista 7	**5**	**4**	**4**	**4**	**4**	**21**
Ej	**35**	**28**	**28**	**32**	**30**	**153**
PP	**5**	**4**	**4**	**4.57**	**4.28**	**21.8**
MP	**5**	**5**	**4**	**5**	**5**	**24**
PA	**100%**	**80%**	**80%**	**91.4%**	**85.7%**	**87.42%**

Figura 9: Resultados obtenidos en la evaluación por el criterio de especialistas.

IP: Importancia de la propuesta.

CM: Contribución al mejoramiento del desarrollo y del producto de la distribución seleccionada para evaluar los procesos propuestos.

CP: Contenido de la propuesta.

AE: Aplicación exitosa de los PIR diseñados en la distribución GNU/Linux Nova.

EP: Evaluación de la propuesta.

PP: Promedio de puntos.

MP: Máxima puntuación.

PA: Por ciento de Aceptación.

3.2.4.1. Cálculo de la concordancia de criterios.

Con el objetivo de obtener resultados válidos se determina la concordancia de criterios utilizando la Prueba de Hipótesis.

Dada la importancia que tiene la evaluación de la propuesta se determinó utilizar el coeficiente de Kendall (W) en la determinación de la concordancia en el criterio de los especialistas. Este estadígrafo tiene mayor rigor matemático que el coeficiente de concordancia (Cc), además permite arribar a un resultado con menos rondas.

Para el cálculo de W se hace uso del programa estadístico Statistical Product and Service Solutions (SPSS), los resultados obtenidos del cuestionario aplicado a los especialistas constituyen la entrada al cálculo de W. Los valores del coeficiente de Kendall "W" deben oscilar entre 0 y 1 ($0 < W < 1$), si W alcanza el valor uno ($W = 1$) entonces existe una concordancia total de criterios, mientras mayor sea el valor de W, es decir, cuanto más se acerque a uno, mayor será la concordancia entre los especialistas. La concordancia se considera aceptable si $W \geq 0.5$.

Luego de obtener los resultados, se utiliza la Prueba de Significación de Hipótesis, planteándose la hipótesis nula (H0) y la alternativa (H1) de la siguiente forma:

H0: no existe concordancia entre los especialistas. $W=0$

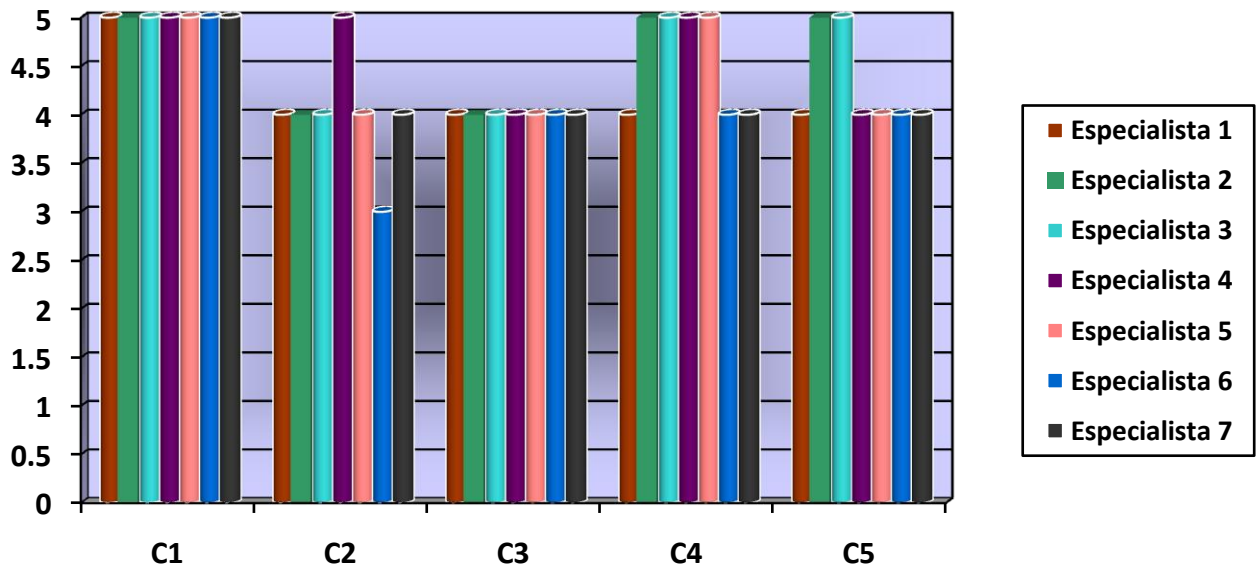
H1: existe concordancia entre los especialistas. $W \neq 0$.

El resultado fue $W=0.6$ por lo que se considera aceptable la concordancia entre los especialistas.

Para realizar el cálculo de la concordancia se debe tener en cuenta el tamaño de la muestra (N), es decir, la cantidad de criterios a evaluar, en este caso $N<=7$ y se utiliza como estadígrafo el numerador del coeficiente de Kendall, nivel de significación (S). Otra vía para verificar que la concordancia es aceptable es comparar el nivel de significación (Asymp.Sig en el SPSS) con 0,05.

Cuando el resultado obtenido es menor que 0.05 se acepta la H1, de lo contrario se acepta H0. En este caso Asymp.Sig = 0,002 < 0,05, quedando evaluados los PIR propuestos.

3.2.5. Conclusiones de la evaluación por el método Delphy.



Gráfica 1: Resumen de la evaluación por el criterio de especialistas

Analizando los resultados obtenidos en la evaluación por el criterio de especialistas en la figura 9 se concluye que las evaluaciones otorgadas por los especialistas son satisfactorias encontrándose entre los 4 y 5 puntos, el por ciento de aceptación está por encima del 87%. Por tanto, de acuerdo al criterio de especialistas en el tema la propuesta responde a los objetivos trazados en la investigación.

3.3. Evaluación por el método experimental.

Para comprobar si algunos de los procesos propuestos ayudan positivamente al desarrollo de software se utiliza el método experimental. Se realiza un análisis del problema existente antes de existir los PIR para el desarrollo de las distribuciones GNU/Linux y luego de insertar los mismos. Los procesos elaborados tienen como meta eliminar todas las deficiencias existentes en cuanto a la administración de requisitos en las distribuciones GNU/Linux. Esta evaluación se llevará a cabo en la distribución Nova. En estos momentos el desarrollo de Nova se encuentra en la fase de mantenimiento por lo que no es posible aplicar todos los procesos. Los procesos que se van a insertar en el desarrollo son

- Descripción de los requerimientos del software.
- Verificación del sistema instalable.

Se hacen necesarios una serie de indicadores que sin duda permitan el mejoramiento de estos

procesos. Para la descripción de los requerimientos del software se evalúa la LRP de los productos Nova Servidores y Nova Escritorio. Estos indicadores constituyen requisitos a tener en cuenta para la ejecución de los procesos como son:

Descripción de los requerimientos del software.

I1: Cumplimiento con las características de los requerimientos.

I2: Descripción del alcance del sistema.

I3: Identificación de los elementos de entradas y salidas.

I4: Definición de los requisitos funcionales y no funcionales.

I5: Definición del equipo que participa en el proyecto.

I6: factibilidad de requisitos.

I7: Complejidad de los requerimientos.

I8: Prioridad para el equipo de desarrollo, teniendo en cuenta las necesidades de los usuarios.

A cada indicador se le asigna un valor entre 0 y 5 los cuales describen hasta qué punto se ejecutan correctamente cada uno de esos indicadores, teniendo en cuenta que 0 significa que no se ejecuta el indicador.

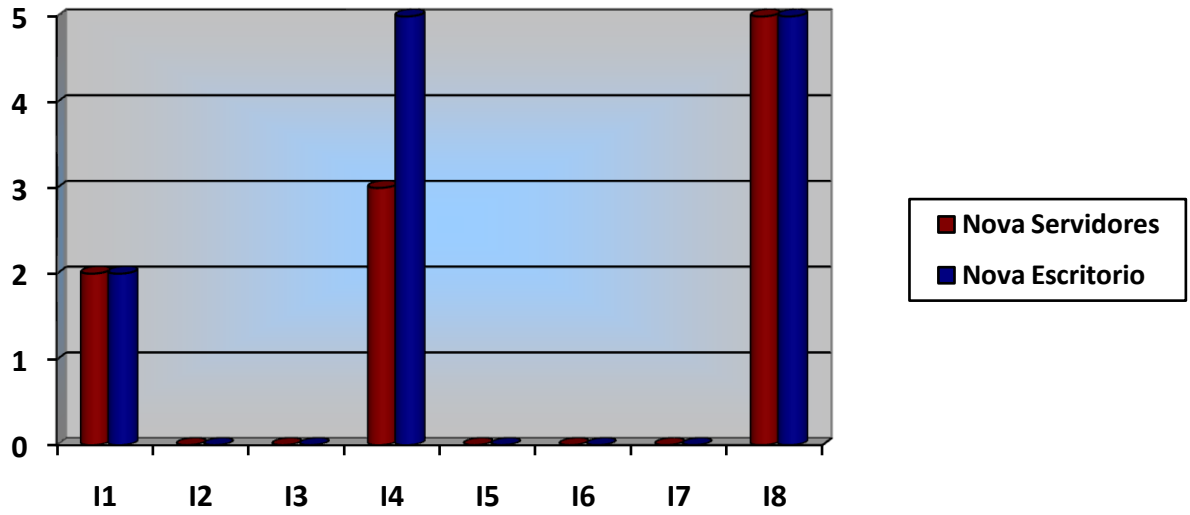
A continuación se muestra como se evidencian tales parámetros en Nova antes y después de insertar estos PIR escogidos en productos como Nova Servidores y Nova Escritorio.

Descripción de los requisitos de Nova Servidores y Nova Escritorio antes de existir el proceso

Nova Servidores	Nova Escritorio
Los requisitos no cumplen con algunas de las características que deben tener como por ejemplo existen algunos que no son claros pues hay palabras que el usuario puede no saber su significado.	**Los requisitos no cumplen con algunas de las características que deben tener como por ejemplo existen algunos que no son claros pues hay palabras que el usuario puede no saber su significado, así como también existen requisitos ambiguos.**
No se describe el alcance del sistema.	**No se describe el alcance del sistema.**
No se definen los elementos de entradas y salidas.	**No se definen los elementos de entradas y salidas.**
Existen los requisitos pero no se clasifican.	**Se clasifican los requisitos.**

No se definen quienes participan en el proyecto por lo que no se conoce quien es el superior en el organigrama del proyecto.	**No se definen quienes participan en el proyecto por lo que no se conoce quien es el superior en el organigrama del proyecto.**
No se puede probar si los requisitos son factibles.	**No se puede probar si los requisitos son factibles.**
No se conoce la complejidad de los requerimientos.	**No se conoce la complejidad de los requerimientos.**
Se conoce la prioridad de los requerimientos.	**Se conoce la prioridad de los requerimientos.**

Antes

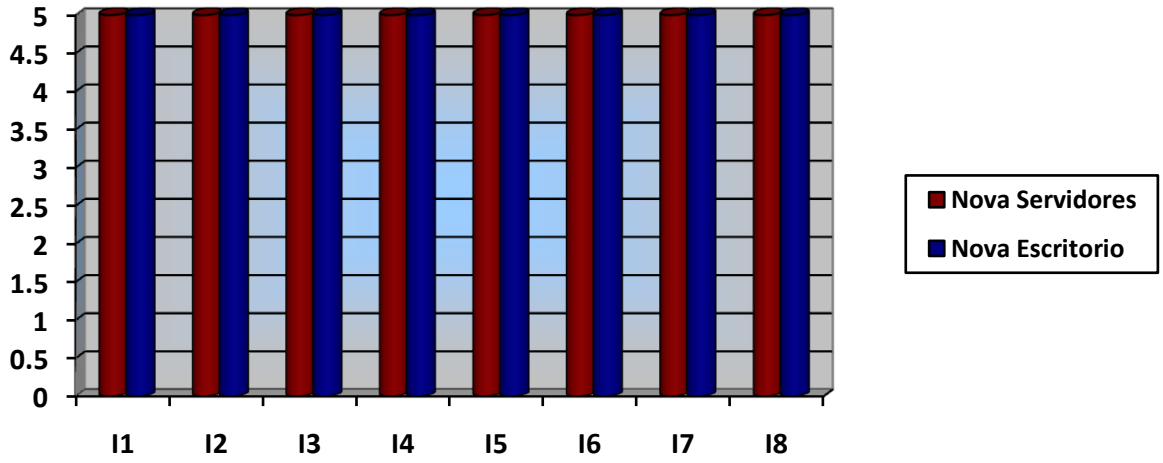


Después

Luego de observar cómo se comportan los criterios establecidos en la ERS que actualmente tiene las variantes de Nova, se lleva a cabo el PIR definido para la descripción de los requerimientos del software que al aplicarlo se puede apreciar el cumplimiento del documento con todas las características de los requerimientos, se describe el alcance del sistema así como los elementos de entrada y salidas, se definen los requisitos funcionales y no funcionales y otras informaciones que sirven de soporte y guías para fases posteriores como por ejemplo, quienes participan en el proyecto conociéndose quien es el superior en el organigrama del proyecto, además de tener

conocimiento de los requisitos factibles, y se establece la complejidad de los requerimientos así como la prioridad.

Después de puesto en práctica este PIR el gráfico de evaluación de los requisitos quedaría de la siguiente forma:



Verificación del sistema instalable.

I1: Documento de ERS libre de errores.

I2: Existencia de los criterios de evaluación de cada una de las características que se deseen revisar.

I3: Funcionamiento de los requerimientos en un escenario de prueba que este en función de los requisitos no funcionales.

I4: Funcionalidad de los requisitos con lo que debe cumplir el producto.

I5: Documentación de las no conformidades detectadas y estrategias para eliminar las mismas.

I6: Seguimiento a las acciones correctivas.

Antes

1. El documento de ERS presenta errores por lo que la verificación del sistema instalable no garantiza que los requerimientos presentes cumplan con la calidad requerida.

2. Están presentes los criterios a evaluar de cada una de las características que se desean revisar pero no están bien especificados.

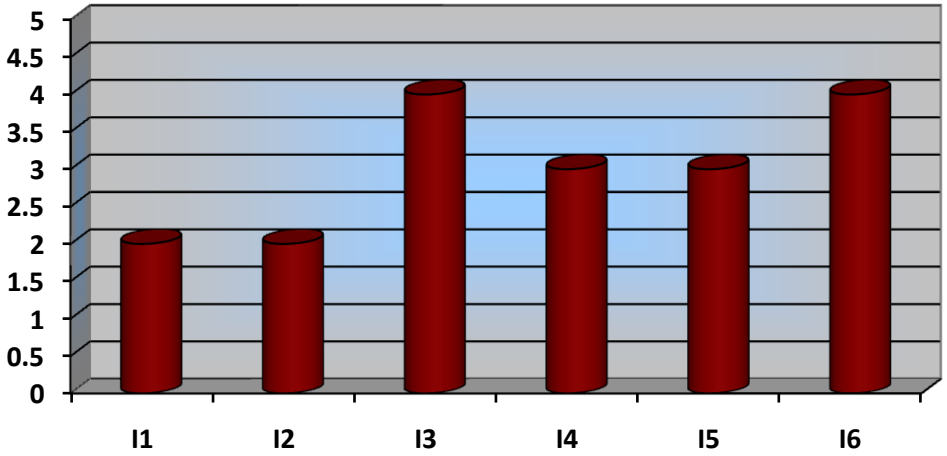
3. El sistema se demora en levantar aproximadamente 30 minutos así como las aplicaciones algunas un tiempo menos que otras pero demoran en cargar, esto ocurre en PC de 256 de

memoria RAM con un microprocesador CELERON 3.0, lector DVD-ROM, Board ACUCA2, y HDD CT80-SATA.

4. No se puede evaluar la funcionalidad de algunos requisitos porque en algunos casos no aparece la opción de la funcionalidad, y en otros casos aparece pero no cumple con lo que realmente el sistema debe hacer. También existe el caso en que funcione un requisito pero no da la posibilidad de realizar alguna acción.

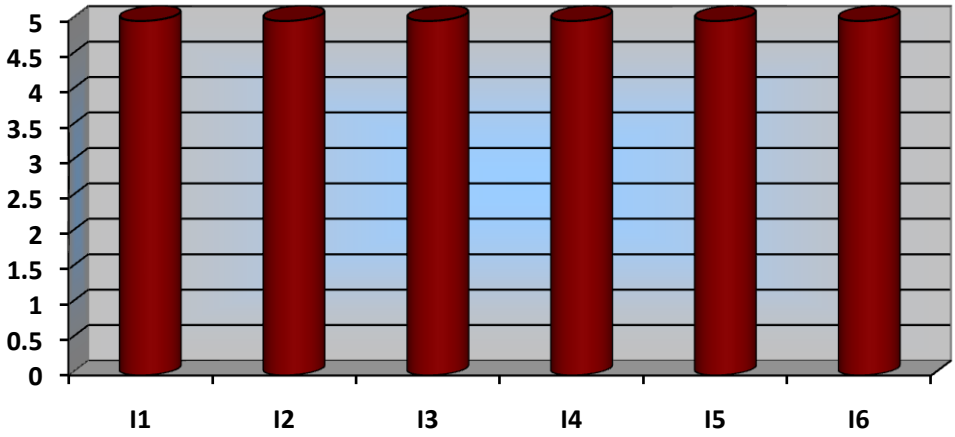
5. Se establecen las acciones correctivas pero no se documentan en una plantilla por lo que se puede caer en el desorden y perder la trazabilidad del proyecto.

6. Se establece el seguimiento a las acciones correctivas pero no se cumple con ello.



Después

Primeramente se cuenta con un documento de ERS libre de errores por lo que la verificación garantiza que los requerimientos presentes sigan los estándares de calidad. Están bien especificados los criterios a evaluar de cada una de las características que se desean revisar, puede o no existir deficiencias en el funcionamiento del sistema en máquinas con las características anteriormente mencionadas así como en la funcionalidad con la que debe cumplir el producto ya que luego de cada iteración de prueba se modificará el producto limando los errores encontrados en las iteraciones anteriores. Se definen y documentan las acciones correctivas además de dar seguimiento a las mismas.


5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
I1
I2
I3
I4
I5
I6

Conclusiones

Luego de haber culminado dicha investigación se arribaron a las siguientes conclusiones:

- Se obtuvo un estudio sobre las tendencias existentes a nivel mundial, en Cuba y en la UCI, así como aspectos relacionados con la IR, tales como: herramientas automatizadas, metodologías de desarrollo que contemplan la IR, sus principales actividades, entre otros. Esta investigación permitió determinar la herramienta a utilizar para la gestión de requisitos en la distribución GNU/Linux Nova.
- Mediante el análisis actual del proceso de desarrollo de la distribución GNU/Linux Nova, en la segunda auditoría interna al proyecto se comprobó que los PIR que se realizan en cada una de las líneas de desarrollo son insuficientes, porque no se tiene en cuenta la administración de los cambios que puedan sufrir los requisitos, el rastreo o trazabilidad de los mismos.
- Se diseña un conjunto de PIR que describen la forma de llevar a cabo la IR en el desarrollo de distribuciones de GNU/Linux contribuyendo a la mejora del PDSW.
- Los PIR diseñados fueron evaluados satisfactoriamente por un grupo de especialistas con experiencias en el tema.
- Se comprobó la propuesta en la distribución GNU/Linux Nova antes y después de existir los PIR en el desarrollo de las distribuciones de GNU/Linux obteniéndose un exitoso resultado.

Recomendaciones

- Aplicar la propuesta a otras distribuciones de GNU/Linux.
- Aplicar los procesos a la propia distribución de GNU/Linux Nova para corregir los errores existentes y perfeccionar el PDSW de la misma.

.

Referencias bibliográficas

1. Quispe-Otazu.,Rodolfo. Computacion e Informatica. *¿Que es la Ingenieria de Requerimientos?* [Online] Agosto 12, 2007. [Cited: Octubre 15, 2010.] http://www.rodolfoquispe.org/blog/page/3/

2. j, Lizka Johany herrera. ilustrados.com. *Ingeniería De Requerimientos - Ingeniería De Software.* [Online] Octubre 9, 2003. [Cited: Noviembre 15, 2010.] http://www.ilustrados.com/publicaciones/EpyVZFEVukfVKPBUot.php

3. Pakoz, Zuñiga. Procesos De La Ingenieria De Requerimientos. [Online] [Cited: Noviembre 27, 2010.] http://www.mitecnologico.com/Main/ProcesosDeLaIngenieriaDeRequerimientos

4. authorstream. *Trazabilidad.* [Online] [Cited: Noviembre 28, 2010.] http://www.authorstream.com/Presentation/aSGuest21435-208628-trazabilidad-aaa-education-ppt-powerpoint/

5. Fernandez-Baillo, David Carrero. Diccionario Informático. [Online] [Cited: Noviembre 27, 2010.] http://www.glosarium.com/term/1283,14,xhtml.

6. programa.us. *musica*. [Online] [Cited: Noviembre 27, 2010.] http://www.programa.us/asesorias/educacion/musica/

7. Orrego, Juan. Efigie del Tiempo. *Que es la Ingeniería.* [Online] Febrero 26, 2005. [Cited: Diciembre 1, 2010.] http://orrego.wordpress.com/?s=La+Ingenier%C3%ADa+como+concepto+y+como+palabra+es+tan+antigua

8. Capítulo 1. IAGP 2005/06. *Ingeniería del software*. [Online] Diciembre 16, 2006. [Cited: Diciembre 2, 2010.] http://www.um.es/docencia/barzana/IAGP/Iagp1.html

9. liderdeproyecto.com. [Online] [Cited: Diciembre 10, 2010.] http://www.liderdeproyecto.com/manual/control_de_cambios_a_los_requerimientos.html

10. Victor Anaya, Patricio Letelier. Trazabilidad de Requisitos Adaptada a las Necesidades del Proyecto. Universidad Politécnica de Valencia: s.n.

11. Gracia, Joaquin. InegenieroSoftware. [Online] [Cited: Enero 12, 2011.] http://www.ingenierosoftware.com/calidad/cmm-cmmi-nivel-2.php.

12. Estándares de calidad- Escuela de Computación e Informática. Universidad de Costa rica, San Pedro: s.n.

REFERENCIAS BIBLIOGRÁFICAS

13. Ingeniería Software. *Proceso de Administración de Requerimientos de RUP.* [Online] Noviembre 28, 2008. [Cited: Enero 15, 2011.] http://clases3gingsof.wetpaint.com/page/Proceso+de+Administraci%C3%B3n+de+Requerimientos+de+RUP

14. Desarrollando aplicaciones informáticas con el Proceso de Desarrollo Unificado (RUP). **Zaragoza, Mtra. María de Lourdes Santiago.** Profesora del Programa Educativo de tecnologías de la Información y Comunicación.

15. Media Jornada "Open UP - Proceso de desarrollo Ágil". Auditorio del MUG: Rivadavia 1479 1er Piso - Ciudad de Buenos Aires: s.n., 2008.

16. slideshare. *Metodologia upen up.* [Online] [Cited: Enero 25, 2011.] http://www.slideshare.net/samith/metodologia-upen-up-3439131

17. Ana Patricia Rodriguez F, Josue Polanko, Darwin hernandez. scribd.com. [Online] [Cited: Enero 25, 2011.] http://www.scribd.com/doc/37116717/Open-Up

18. Extreme Programming. *Exposición -xp-1*. [Online] [Cited: Febrero 5, 2011.]

19. Chaves, Michael Arias. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. [Online] Julio 7, 2006. [Cited: Febrero 6, 2011.] http://www.latindex.ucr.ac.cr/intersedes10/10-art_11.pdf

20. Entregable D3. [Online] [Cited: Febrero 26, 2011.] http://docs.google.com/viewer?a=v&q=cache:Am0B3F2rC2gJ:www.ines.org.es/vulcano/wp-content/uploads/2009/05/d33-v-13.pdf

21. Requerimentone. *ReqMan Introduction.* [Online] [Cited: Febrero 15, 2011.] http://www.requirementone.com/Free_project_management_tool/Our_product/ReqMan_introduction.aspx

22. Beekmans, Gerard. *Linux From Scratch: Versión 6.3*.s.l.:Copyright © 1999–2007 Sobre el texto original: Gerard Beekmans.

Bibliografía

1. **Pfleeger Shari Lawrence**, *Ingeniería de software, teoría y práctica,* México, Prentice Hall, 2002
2. **F.Brooks.** *The.Mythical.Man.Month.*s.l.: ISBN 0-201-83595-9, 2008
3. **Sommerville, Ian.** *Ingeniería de software.* México: 6ta Edición Addison Wesley
4. **Sommerville, Ian.** *Ingeniería del software.* s.l.: 7ma Edición, Prentice hall, 2005
5. **S, Pressman Roger.** *Ingeniería del Software, Un enfoque práctico.* México: 6ta Edición, 2003