UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 1, Centro de Software Libre

Procedimiento de construcción de imágenes de la personalización de GNU/Linux Nova Escritorio.

Trabajo final presentado en opción al título de

Máster en Informática Avanzada

Autor: Manuel Enrique Peiso Cruz

Tutor: Dr C Omar Correa Madrigal

La Habana, noviembre de 2018

AGRADECIMIENTOS

A mi familia por su apoyo incondicional.

A Leo y el resto de mis amigos por estar siempre presentes.

A todos los que colaboraron.

DEDICATORIA

A mi mamá, papá y abuela.

A mis hermanos.

DECLARACIÓN JURADA DE AUTORÍA

Declaro por este medio que yo Manuel Enrique Peiso Cruz, con carnet de identidad 91081115944, soy el autor principal del trabajo final de maestría PROCEDIMIENTO DE CONSTRUCCIÓN DE IMÁGENES DE LA PERSONALIZACIÓN DE GNU/LINUX NOVA ESCRITORIO, desarrollada como parte de la Maestría de Informática Avanzada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

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

RESUMEN

Las imágenes constituyen un elemento fundamental de las distribuciones de GNU/Linux, representando la vía oficial para la instalación del sistema en una estación de cómputo. Actualmente en la distribución cubana de GNU/Linux Nova Escritorio la falta de estandarización en los procesos vinculados a las imágenes atentan contra la adecuación funcional y la socio-adaptabilidad del producto, lo que se ve reflejado en el servicio de migración que se presta a los Organismos de la Administración Central del Estado. La presente investigación realiza un estudio de la manera en que otras distribuciones realizan las imágenes, determina un conjunto de problemáticas que deben ser corregidas y propone un procedimiento para la creación de las imágenes de Nova Escritorio teniendo en cuenta la instalación en modo UEFI y la creación de memorias de arranque con usb-creator. El procedimiento se define teniendo en cuenta que el proceso de migración constituye la principal retroalimentación para el mejoramiento del producto. La implementación e implantación del resultado de la investigación aumenta la adecuación funcional y la socio-adaptabilidad de la distribución cubana de GNU/Linux Nova Escritorio.

Palabras clave: imágenes, Linux, distribución, procedimiento, migración.

ABSTRACT

The images become a fundamental element of the distributions of GNU / Linux, representing the official way for the installation of the system in a computer station. Currently in the Cuban distribution of GNU / Linux Nova Desktop lack of standardization in the processes linked to the images undermine the functional adequacy and the socio-adaptability of the product, which has been reflected in the service of migration that is provided to the Organisms of the Central State Administration. The present investigation carries out a study of the way in which other distributions, the images, the determination, the set, the problematic, the correction, the task, the work, the writing, the account, the installation, the way, the creation, application, mode, creation, employment and creation. of boot memories with usb-creator. The procedure is defined taking into account that the migration process is the main feedback for the improvement of the product. The implementation and implantation of the research result increases the functional adequacy and socioadaptability of the Cuban distribution of GNU / Linux Nova Desktop.

Keywords: images, Linux, distribution, procedure, migration.

ÍNDICE

INTRODUCCIÓN

En la informática moderna el software libre se ha destacado por suponer un nuevo paradigma en el desarrollo de software y su comercialización. Un desarrollo colaborativo en pos del bien común y no del enriquecimiento personal ha contribuido a romper la hegemonía de los sistemas privativos. Estos últimos constituyen aplicaciones que no pueden ser modificadas y distribuidas libremente, usualmente se comercializan a precios muy elevados y suelen utilizar licencias de software que deben ser compradas cada determinado período de tiempo para continuar su uso (Stallman, 2002).

El interés de obtener un producto aunando el conocimiento de un conjunto de personas para lograr un sistema con calidad, sin importar ubicación geográfica ni intereses materiales y que permitiera su libre modificación, estudio y distribución pasó a ser fuente de motivación para un conjunto de personas que creían que las tecnologías de la información y las comunicaciones debían ser accesible para todos. Un ejemplo fehaciente de desarrollo colaborativo y libre es GNU/Linux, un proyecto que se dio a la tarea de crear un sistema operativo que cumpliera con las libertades antes mencionadas.

GNU/Linux se ha destacado por su estabilidad de operación, su velocidad y su capacidad para la administración eficiente de los recursos de la computadora, como la memoria, la unidad central de procesamiento y el espacio en disco. Todas estas características lo han llevado a diferenciarse del resto de los sistemas operativos del mundo y ha propiciado que en la actualidad las distribuciones de GNU/Linux hayan alcanzado una significativa popularidad ampliando su uso a varias esferas de la sociedad y convirtiéndose en una alternativa a nivel mundial (Stallman, 2002).

En América Latina varios países han optado por esta opción; el gobierno de Venezuela estableció por decreto presidencial el uso preferente de software libre y de GNU/Linux en toda la administración pública, incluso en los ministerios y oficinas gubernamentales (Chávez, 2004). En ese país se desarrolla la distribución GNU/Linux Canaima, usada en el proyecto Canaima Educativo y en la producción de computadoras de escritorio a bajo precio (Gavasa, 2014). En 2008 el gobierno de Ecuador mediante el Decreto 1014 estableció la obligatoriedad del uso de tecnologías de código abierto en entidades públicas (Correa, 2010). Uruguay aprobó en diciembre de 2013 la Ley de Software Libre y Formatos Abiertos en el Estado, la cual establece la obligatoriedad del uso de estos formatos en las instituciones gubernamentales, además prioriza el desarrollo en software libre para las soluciones informáticas (El Senado y la Cámara de Representantes de la República Oriental del Uruguay, 2014).

Cuba también se interesó en el software libre y uno de los pasos que tomó fue la creación de un sistema operativo nacional basado en GNU/Linux. A este sistema se le denominó GNU/Linux Nova (Nova en lo adelante) y desde su creación se convirtió en un producto que podía utilizarse, en caso de ser necesario, para sustituir a Microsoft Windows, el sistema operativo privativo usado por

excelencia en la industria cubana; un sistema muy caro y que incluye varios softwares que también son muy costosos. El desarrollo de este producto fue tarea del Centro de Software Libre de la Universidad de las Ciencias Informáticas (Nova, 2017).

Con posterioridad quedó establecido en el lineamiento 108 de la Política Económica y Social del Partido y la Revolución el interés de fomentar el uso de las nuevas tecnologías de la información y las comunicaciones (Partido Comunista de Cuba, 2016), lo que impuso el reto a los desarrolladores e investigadores en la rama informática de buscar soluciones de avanzada para fomentar el proceso de informatización de la sociedad, donde nuevamente la distribución cubana de GNU/Linux se avizoraba como una solución viable. Nova es una distribución que utiliza el núcleo de Linux e incluye un amplio paquete de aplicaciones informáticas dedicadas a satisfacer las necesidades de migración a plataformas y estándares abiertos que experimenta Cuba como parte del proceso de informatización de la sociedad (Nova, 2017).

Nova es desarrollado por los cubanos y para los cubanos, siendo esta una de sus mayores fortalezas pues permite la correspondencia entre las políticas de informatización nacional y la socio-adaptabilidad del sistema. El amplio espectro y diversidad de las estaciones de trabajo que conviven en el entorno tecnológico cubano crea la necesidad de un sistema capaz de adaptarse a este escenario. Por ello, la distribución cubana de GNU/Linux se divide en 3 productos fundamentales: Nova Ligero, para estaciones de trabajo con bajas prestaciones de hardware (menos de 512Mb de memoria RAM); Nova Escritorio, para estaciones de trabajo con buenas prestaciones de hardware (de 512Mb en adelante) y Nova Servidores para ordenadores que tienen como función el control de redes, centro de datos, entornos empresariales y de desarrollo, entre otros (Nova, 2017).

El proceso de construcción de Nova Escritorio y de todas las personalizaciones a la medidas que necesitan determinados organismos e instituciones del país se divide en varios momentos que están enfocados en mantener altos niveles de excelencia en los aspectos de seguridad, socio-adaptabilidad, soberanía tecnológica y sostenibilidad (Pierre, 2015). Estos momentos son: construcción de un repositorio básico que permita la creación del sistema base, construcción de un entorno de escritorio que permita el trabajo gráfico en el sistema, la modificación de aplicaciones para arreglar errores detectados por el equipo de desarrollo y para integrarlas al entorno de escritorio desarrollado. De forma transversal se construyen imágenes del sistema que es el producto instalable que permite verificar el correcto funcionamiento de lo desarrollado hasta el momento. Cada una de ellas es esencial y su correcta ejecución es fundamental en la calidad del producto final.

Precisamente la calidad del producto es primordial pues de ello depende la aceptación por parte del usuario que muchas veces se siente renuente a la hora de migrar a este tipo de tecnologías. Parte esencial de la calidad del producto es la calidad de cada uno de los procesos que intervienen en su desarrollo (Organización Internacional de Estandarización, 2008), a pesar de ello la construcción de

las imágenes de Nova es actualmente uno de los problemas que influyen en su calidad como producto. El modelo para la construcción de la distribución no está completo y no cubre todas las aristas del proceso. Un ejemplo es la poca estandarización de las herramientas de construcción que implica que los atributos de un producto pueden variar en dependencia de las herramientas utilizadas por los desarrolladores.

La creación de las imágenes es un procedimiento que, a pesar de realizarse utilizando herramientas informáticas, resulta muy complicado y no se encuentra definido correctamente, sino que se utiliza como guía los pasos propuestos por Ubuntu en su página oficial. El mismo se realiza utilizando una consola Linux y un enorme conjunto de comandos que son necesarios dominar y que ante cualquier imprecisión pueden dar al traste con el trabajo realizado. El tiempo que se tarda en realizar una imagen varía en dependencia de la complejidad que las mismas requieran, pero incluso las más sencillas requieren de procesos largos y de gran consumo de hardware para la estación de trabajo. No existe tampoco ninguna guía oficial en la que quede plasmado el mecanismo para su realización, por lo que constituye un conocimiento tácito que no se encuentra explicitado, esto provoca que no todas las imágenes realizadas poseen las mismas características y algunas presentan errores por incompatibilidad de las herramientas utilizadas para su construcción con otras imprescindibles para su correcto funcionamiento. En el proceso de construcción de la personalización, cada vez que se detecta una no conformidad es necesario realizar una imagen nueva, por tanto, es un procedimiento que se realiza muchas veces desde el inicio del desarrollo hasta que se libera el producto final.

Una vez concluidas las imágenes se le realizan pruebas exploratorias y pruebas a cada uno de los requisitos teniendo en cuenta los casos de prueba diseñados por los integrantes del proyecto, pero no existen pruebas dentro del proceso de construcción de las imágenes lo cual provoca que el producto final en ocasiones contenga errores que pudieron ser detectados en etapas tempranas. El repositorio de aplicaciones se actualiza constantemente, pero debido a la poca integración entre estos dos procesos muchas veces el producto sale con versiones de aplicaciones atrasadas lo cual provoca inconformidades por parte del usuario final que detecta errores que ya fueron corregidos, pero no actualizados en la imagen.

Como se planteó anteriormente, para la construcción de las imágenes de Nova se siguen los pasos definidos por Ubuntu, a pesar de ello, y aunque Ubuntu permite la instalación en discos duros con tablas de particiones GPT, no ocurre así en el caso de Nova, impidiendo la convivencia del sistema con las últimas versiones de sistema operativo Windows, aún presente en la mayoría de los computadores personales y de muchas estaciones de trabajo que son producidas con discos duros que solo poseen este tipo de tablas de particiones. Esto resulta contradictorio si se tiene en cuenta que se viola el principio de socio-adaptabilidad definido por Pierre y que constituye uno de los preceptos fundamentales de la distribución. En la última versión de Nova, las imágenes obtenidas resultaron incompatibles con la herramienta oficial de Nova para la creación de discos de arranque

impidiendo la realización de memorias de arranque desde la misma y obligando la utilización de aplicaciones de terceros para satisfacer esta problemática.

Estos elementos afectan directamente la adecuación funcional del producto, si se tiene en cuenta que varios de los requisitos funcionales definidos y firmados por el cliente no se realizan o presentan dificultades, fundamentalmente en el proceso de instalación y de construcción de memorias de arranque. Por otra parte, el proceso de migración a software libre y código abierto que experimenta Cuba también se ve afectado, pues la manera en la que se realiza la construcción de imágenes en la actualidad provoca que el especialista de migración debe llevar consigo tres memorias de arranque como mínimo con las distribuciones que se utilizan en las estaciones de trabajo comunes: Nova Escritorio i386, Nova Escritorio amd64 y Nova Ligero. En dependencia de las características y exigencias del cliente, se utiliza la distribución que más se ajuste. Esto afecta el aprovechamiento de las capacidades de almacenamiento de los dispositivos externos, cuya capacidad suele ser, como mínimo, de ocho gigabytes y de los cuales solo se utiliza alrededor de 1,5 gigabytes por concepto de sistema operativo. También resulta muy complejo para el especialista de migración tener que portar tres dispositivos externos y un gasto innecesario de recursos.

A partir de la situación problemática planteada surge el siguiente **problema científico**: ¿Cómo aumentar los niveles de adecuación funcional y socio-adaptabilidad de las imágenes de la personalización de GNU/Linux Nova Escritorio?

El **objeto de estudio** se centra en las imágenes de personalizaciones de GNU/Linux, teniendo como **campo de acción** las imágenes de las personalizaciones de GNU/Linux Nova Escritorio.

El **objetivo general** de la investigación consiste en elaborar un procedimiento de construcción de imágenes en la personalización cubana de GNU/Linux Nova Escritorio, que tenga en cuenta la instalación en estaciones de trabajos de última generación y la creación de memorias de arranque para el proceso de migración para aumentar la adecuación funcional y la socio-adaptabilidad del producto final.

Para solucionar el problema y como resultado del análisis de la literatura especializada, se formuló la siguiente **hipótesis**: la elaboración de un procedimiento de construcción de imágenes en la personalización cubana de GNU/Linux Nova Escritorio, que tenga en cuenta la instalación en estaciones de trabajos de última generación y la creación de memorias de arranque para el proceso de migración, contribuirá a aumentar la adecuación funcional y la socio-adaptabilidad del producto final.

El objetivo general fue desglosado en los siguientes **objetivos específicos:**

1. Construir el marco teórico y referencial de la investigación, relacionado con el desarrollo de imágenes en personalizaciones de GNU/Linux.
2. Describir el procedimiento de construcción de imágenes en la personalización cubana de

GNU/Linux Nova Escritorio.

3. Aplicar el procedimiento de construcción de imágenes en la personalización cubana de GNU/Linux Nova Escritorio.
4. Validar la propuesta a través de los métodos definidos en la investigación.

En el proceso investigativo se utilizaron los métodos de investigación que a continuación se exponen.

Métodos teóricos:

- **Hipotético – deductivo:** para formular la hipótesis de la solución y proponer nuevas líneas de trabajo a partir de la solución.
- **Analítico – sintético:** para descomponer el problema de investigación en elementos, profundizar en su estudio y luego sintetizarlos en la solución propuesta.

Métodos empíricos:

- **Análisis documental:** en la revisión de la literatura especializada, tanto académica como empresarial, para extraer la información necesaria que permitió realizar el proceso de investigación.
- **Entrevista a profundidad:** en la obtención de información a partir de la experiencia de especialistas acerca de la construcción de personalizaciones de GNU/Linux.
- **Análisis comparativo:** para detectar similitudes, diferencias e insuficiencias en los procedimientos de construcción de imágenes de Ubuntu con los de Nova.
- **Controles de aplicabilidad:** para validar con los profesionales la relevancia y aplicabilidad del procedimiento de construcción de imágenes en la personalización cubana de GNU/Linux Nova Escritorio.
- **Métodos estadísticos:** en el análisis de las encuestas aplicadas a expertos y potenciales usuarios.

Estructura de la Tesis

La presente tesis está compuesta por introducción, tres capítulos, conclusiones, recomendaciones y bibliografía. Se adjuntan anexos que contribuyen a su comprensión.

- **Capítulo I:** Dedicado a la caracterización de las imágenes de la distribución cubana de GNU/Linux Nova Escritorio, los procedimientos de construcción de imágenes de GNU/Linux existentes, las pruebas que se realizan y las deficiencias durante el proceso de instalación y de creación de memorias de arranques durante el proceso de migración.
- **Capítulo II:** Aborda la concepción del procedimiento. Se diseña, se describe y se implementa la

propuesta de solución.

- **Capítulo III:** Contiene la evaluación de los resultados alcanzados en la investigación los que permitieron confirmar la factibilidad, pertinencia y contribución de la propuesta de solución al procedimiento de construcción de imágenes en la personalización cubana de GNU/Linux Nova Escritorio.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan los conceptos fundamentales que sustentan la presente investigación. Se realiza un resumen de los procesos que se relacionan con la construcción de imágenes de las personalizaciones de GNU/Linux y cómo se manifiestan en Nova, así como las ineficiencias dentro de la construcción de las mismas para determinar los aspectos a tener en cuenta en la propuesta de solución. Se analizan las pruebas que se le realizan a las imágenes para detectar ineficiencias.

1.1 Definiciones de interés

1.1.1 Linux

Linux es el núcleo, el programa que asigna los recursos de hardware que necesita otro programa para funcionar. El núcleo es una parte esencial del sistema operativo, pero inútil por sí mismo (R. M. Stallman, 2002).

Técnicamente hablando, Linux es un verdadero núcleo Unix[1], a pesar de que no es un sistema operativo Unix porque no incluye todas las aplicaciones Unix, como las utilidades del sistema de fichero, sistema de ventanas y entorno gráfico, sistema de administración de comandos, editores de textos, compiladores, etc. De todas maneras, como la mayoría de estos programas están disponibles libremente bajo la licencia GPL[2], ellos pueden ser instalados en cualquier sistema basado en Linux (Fallis, 2013).

Teniendo en cuenta estas definiciones en la presente investigación se considera a Linux como el corazón del sistema. Se encarga de arrancarlo, gestionar los recursos de la máquina en forma de gestión de procesos, memoria y entrada/salida.

1.1.2 GNU/Linux

GNU/Linux es un sistema operativo, consiste básicamente en GNU[3] con Linux incluido. Todas las llamadas distribuciones de Linux son realmente distribuciones de GNU/Linux (R. M. Stallman, 2002). GNU/Linux es un clon del sistema operativo privativo UNIX que utiliza el núcleo Linux y un conjunto de aplicaciones desarrolladas por el proyecto GNU. Se basa en varias libertades: usar el programa con cualquier propósito, estudiar cómo funciona el programa y modificarlo, adaptándolo a las necesidades del usuario, distribuir copias del programa y mejorarlo y hacer públicas esas mejoras a los demás (Baig & Aulí, 2003).

Por lo que se concluye que GNU/Linux es un sistema operativo que incluye el núcleo de Linux así como un conjunto de aplicaciones desarrolladas por el proyecto GNU, permitiendo la interacción de

---

[1] Unix: Sistema operativo propietario (R. Stallman, 2004).
[2] GPL: es una licencia de derecho de autor ampliamente usada en el mundo del software libre (R. Stallman, 2004).
[3] GNU: sistema de software completamente compatible con Unix (R. Stallman, 2004).

los usuarios con los recursos de un ordenador.

1.1.3 Distribuciones de GNU/Linux

Una distribución de GNU/Linux es una colección de software que forma un sistema operativo basado en el núcleo Linux. Las mismas utilizan una versión específica del núcleo Linux, un formato de empaquetado determinado para gestionar sus aplicaciones, así como un conjunto de aplicaciones, en versiones determinadas, que garantizan el correcto funcionamiento de todo el sistema (Jorba & Suppi, 2004).

Las distribuciones de GNU/Linux no son más que el núcleo de Linux empaquetado, junto a utilidades como la consola bash, un gestor de ventanas, un entorno de escritorio y el compilador GCC, más otra serie de aplicaciones como, por ejemplo, procesadores de texto, reproductores multimedia y navegadores web. En la presente investigación se adopta la definición de distribuciones GNU/Linux de Jorba y Suppi.

1.2 UEFI

La Interfaz de firmware[4] extensible (EFI, por sus siglas en inglés) o su variante de versión 2.x, EFI Unificado (UEFI, por sus siglas en inglés) es un tipo de firmware que está muy extendido en equipos recientes, especialmente los más recientes que 2010. Pretende reemplazar el firmware de BIOS tradicional que prevalece en máquinas anteriores. La mayoría de las computadoras modernas son compatibles con el modo UEFI y el modo BIOS. El modo de inicio que se debe usar depende de varios factores:

1. Coincidencia de modo de arranque: si se tiene un arranque dual con otro sistema operativo, los dos modos de inicio de los sistemas operativos deberían coincidir. La mayoría de las computadoras que incluyen Windows 8 y posterior usan UEFI para arrancar ese sistema operativo, por lo que esta configuración exige el uso del modo UEFI al instalar y arrancar Ubuntu o cualquier distribución de GNU/Linux.
2. Soporte de hardware: algunos dispositivos de hardware funcionan mejor en un modo u otro (por lo general, el modo BIOS es mejor admitido). Este tipo de problema es cada vez menos común.
3. Problemas del gestor de arranque: a veces, un gestor de arranque para un modo funciona mejor que un gestor de arranque para otro modo. Los cargadores de arranque en modo BIOS están mejor probados y, por lo tanto, es menos probable que planteen problemas (Hobson, 2017).

El EFI proporciona un nuevo modelo para la interfaz entre los sistemas operativos y el firmware de

[4] Firmware: es un programa informático que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo.

la plataforma y comprende tablas de datos que contienen información relacionada con la plataforma y llamadas al servicio de arranque en tiempo de ejecución que están disponibles para el sistema operativo y su cargador. Para comunicarse con el EFI, el sistema operativo debe configurarse específicamente para su uso con el EFI, o se debe desarrollar otro software apropiado para permitir dicha comunicación.

Aunque la industria de las computadoras personales (PC, por sus siglas en inglés) está haciendo la transición a sistemas basados en EFI, al menos durante los próximos años habrá muchas personas que seguirán usando sistemas operativos heredados que no pueden comunicarse con el EFI. En consecuencia, se necesita una solución que admita tanto BIOS heredado como EFI (González, 2007).

Para instalar Ubuntu en modo UEFI:

1. Usar un disco de 64 bits de Ubuntu (Ubuntu 32 bits no se puede instalar fácilmente en modo UEFI).
2. Es posible que se desee utilizar una imagen solo de EFI para evitar problemas al arrancar la imagen por error e instalar Ubuntu en el modo BIOS.
3. Usar una versión compatible de Ubuntu. El soporte para UEFI apareció en 11.10, pero se ha vuelto más confiable en las próximas versiones. El soporte para UEFI SecureBoot[5] (arranque seguro) apareció en 12.10 y 12.04.2.
4. Configurar el firmware (BIOS) para iniciar el disco en modo UEFI (Ubuntu, 2015).

Entonces:

No se requiere nada especial si se usa el instalador automático de Ubuntu ("Instale Ubuntu junto a otros" o "Borre el disco e instale Ubuntu").

Si se usa el particionamiento manual ("Algo más"), la diferencia es que se deberá establecer el punto de montaje /boot/efi en la partición UEFI (ESP). Y si no hay ninguna partición UEFI en el HDD, primero habrá que crearla. Tamaño mínimo 100 Mb, formato FAT32, necesita una bandera de arranque.

A pesar de la necesidad imperante de que Nova permita la instalación en modo UEFI para que pueda convivir con las últimas versiones de Windows y teniendo en cuenta que desde la versión 11.10 de Ubuntu apareció el soporte para UEFI, se comprueba que ninguna de las versiones de Nova permite esta opción a pesar de seguir los pasos definidos por Ubuntu para la construcción de las imágenes. Esto viola uno de los principios de la distribución, la socio-adaptabilidad, pues se queda atrasado tecnológicamente hablando e impide la convivencia de Nova con las versiones de

[5] UEFI SecureBoot: es una nueva característica de UEFI que previene de correr un sistema operativo desconocido.

Windows 8 y 10 que ya vienen comúnmente instaladas con este firmware.

En el proceso de instalación modo UEFI el instalador hace uso de un conjunto de aplicaciones que se encuentran almacenadas en un repositorio de código binario interno en el ISO. Estas aplicaciones son:

- b43-fwcutter: este paquete proporciona una herramienta para extraer el firmware de chip inalámbrico BCM43xx de los controladores propiedad de Broadcom.
- bcmwl-kernel-source: este paquete contiene el controlador inalámbrico Broadcom 802.11.
- dkms: proporciona soporte para la instalación de versiones suplementarias de módulos del núcleo.
- libc6-i386: incluye versiones compartidas de la biblioteca C estándar y la biblioteca matemática estándar, así como muchas otras.
- grub: es un gestor de arranque destinado a unificar la carga de arranque en los sistemas operativos X86. Implementa el estándar de arranque múltiple.
- grub-efi: se trata de un paquete de transición ficticio que depende de grub-efi-amd64.
- grub-efi-amd64: es un gestor de arranque portable y potente. Contiene una versión de grub que se ha creado para su uso con arquitectura EFI-AMD64.
- grub-efi-amd64-bin: constituye una dependencia de grub-efi-amd64.
- grub-efi-amd64-signed: este paquete contiene una versión de grub diseñada para su uso en arquitectura EFI-AMD64 firmada con la clave de firma UEFI de Canonical.
- intel-microcode: contiene microcódigo de procesador de sistema actualizado para procesadores Intel 686 e Intel X86-64.
- iucode-tool: es un programa para manipular las colecciones de microcódigos del procesador Intel X86 y X86-64 y para usar las funciones del núcleo para actualizar el microcódigo en los procesadores del sistema Intel.
- lupin: archivos de soporte para instalaciones de montaje en bucle.
- mokutil: este programa proporciona los medios para inscribir y borrar claves del propietario de la máquina almacenados en la base de datos shim.
- mouseemu: es un demonio para emular los botones del mouse.
- setserial: controla la configuración de los puertos seriales.
- shim: gestor de arranque para cargar cargadores de arranque firmados bajo arranque seguro. Su propósito es permitir que un pequeño binario que cambia poco frecuentemente, sea firmado por UEFI.
- shim-signed: este paquete contiene la versión del binario del gestor de arranque firmado por

el Microsoft UEFI CA[6].

- oem-config: realiza la configuración del usuario final después de la instalación inicial.
- oem-config-gtk: interaz GTK para oem-config.
- oem-config-slideshow-ubuntu: presentación de diapositivas diseñada para la configuración de la instalación.
- wvdial: marcador inteligente de protocolo punto a punto para facilitar la configuración de acceso telefónico.
- libuniconf4.6: bibliotecas de red C++ para el desarrollo rápido de aplicaciones.

Como se puede apreciar varias de ellas tienen relación directa con UEFI, especialmente las relacionadas con el cargador de arranque. Nova no posee una versión de binario de gestor de arranque firmado por Microsoft UEFI CA por lo que se puede deducir que esta puede ser una de las causas que imposibilitan el arranque en este modo del sistema (Microsoft, 2014). Pero acaso ¿Ubuntu posee una versión de binario de gestor de arranque firmado por Microsoft UEFI CA para cada una de sus versiones, como Kubuntu, lubuntu, etc.?

El estudio del paquete de código fuente grub-common, responsable de la generación de los paquetes de código binario relacionados con el grub permitió determinar que no es así. Ubuntu camufla varias de sus distribuciones haciéndolas pasar por ella misma de forma tal que todas puedan utilizar el mismo gestor de arranque firmado. Por tanto, se puede concluir que siguiendo esta misma lógica se podría camuflar a Nova de la misma forma y esto permitiría el correcto funcionamiento del sistema en modo UEFI.

Por otro lado, si se analiza el comando utilizado en Nova para la construcción de la imagen:

mkisofs –r –V "$IMAGE_NAME" –cache-inodes –J –l –b isolinux/isolinux.bin –c isolinux/boot.cat –no-emul-boot –boot-load-size 4 –boot-info-table –o nova.iso .

Se puede determinar que carece de los parámetros necesarios para la compatibilidad con UEFI como:

- -e para establecer el nombre de la imagen de arranque EFI.
- -isohybrid-gpt-basdat para establecer la imagen de arranque de El Torito[7] como datos básicos en GPT.
- -isohybrid-mbr para establecer la imagen como híbrida, o sea que funcione tanto para BIOS como para EFI.

[6] Microsoft UEFI CA: certificadora de binarios del grub para UEFI (Microsoft, 2014).
[7] El Torito: es un estándar para crear medios ópticos de arranque como DC-ROM, DVD o BD. Permite ofrecer en las mismas imágenes de arranque alternativas del sistema de archivo ISO para PC-BIOS y para EFI (UEFI, 2014).

Esto también permite determinar que la carencia de estos parámetros impide la compatibilidad del sistema con este firmware.

1.3 Creador de discos de arranque

El Creador de discos de arranque es la herramienta utilizada en Nova para la creación de memorias de arranque. La misma permite, según la descripción que provee su página oficial, la creación de memorias de arranque de Ubuntu, Debian, Gentoo y otras tantas distribuciones de GNU/Linux, pero no funciona para Fedora, CentOS, SolusOS, etc. (Usb-creator, 2018).

Hasta la versión 5.0 de Nova, la herramienta era completamente funcional (versión 0.2.56.3ubuntu0.1) y a través de su uso se generaban todas las memorias de arranques de la distribución, pero con el lanzamiento de Nova 6.0 (versión de usb-creator 0.3.2.1nova1) las memorias de arranques creadas no permitían la instalación del sistema. Un análisis del código fuente de la aplicación permitió detectar un cambio radical en su manera de generar estas memorias. Mientras en Nova 5.0, luego de descomprimir el sistema en el disco duro, instala un cargador de arranque, para lo cual necesita permiso de administración, en Nova 6.0 descomprime el sistema en la estación de trabajo, pero no instala el cargador de arranque. Esto provoca que el sistema no es capaz de levantar la ventana de arranque del sistema (isolinux) e imposibilita el acceso al LiveCD.

A pesar de ello con una imagen de Ubuntu las memorias de arranque del sistema son creadas correctamente, lo que hace pensar que la construcción de la imagen carece de algún tipo de parámetro que posibilite su correcta integración con la nueva versión de usb-creator o que las herramientas utilizadas no son las adecuadas para permitir esto. Primeramente, se supondrá que Ubuntu no utiliza la misma herramienta para la construcción de las imágenes, se buscarán herramientas similares y se aplicará la metodología de prueba y error en busca de una posible solución al problema. Debian, en su wiki oficial propone la utilización de la herramienta xorriso sobre genisoimage (mkisofs) debido a que esta última no incluye nuevas características ni da soporte para la corrección de errores (Debian, 2018). Una vez probado el proceso con esta nueva herramienta se logró comprobar el correcto funcionamiento del creador de imágenes de discos para una primera y satisfactoria prueba, por lo que se puede concluir que el cambio de genisoimage por xorriso soluciona el problema detectado en la creación de imágenes de disco de la distribución cubana de GNU/Linux Nova.

1.4 Procedimiento de construcción de imágenes

El procedimiento de construcción de imágenes de Nova es inexistente. Se heredan los pasos definidos por Ubuntu y Debian. Ubuntu se basa en Debian y Nova se basa en Ubuntu, por ellos muchos de los pasos en su proceso de desarrollo son similares. Para darle identidad a Nova se modifican un conjunto de paquetes que lo caracterizan, donde se define el nombre de la distribución,

scripts de configuraciones, entorno de escritorio, metapaquetes[8] de aplicaciones que responden a los requisitos definidos por el cliente, etc.

El estudio de la bibliografía especializada permitió identificar que Linux from scratch (Beekmans, 2010) describe cada uno de los pasos a tener en cuenta para la construcción de una distribución de GNU/Linux. En él se definen las aplicaciones y el orden en que deben ser compiladas para crear un sistema base, así como cada uno de los elementos a tener en cuenta para lograr una imagen que se adapte a las características del usuario que desarrolla la distribución. A pesar de ello es un proceso extremadamente complicado y lento, además de que hay aplicaciones como debootstrap que automatizan los elementos más complejos explicados en este libro.

Fue detectado, en la página oficial de Ubuntu, un conjunto de pasos para crear una imagen y que se supone, sea el utilizado en las distribuciones que lo utilizan como base. Como es lógico el mismo está enfocado a las características específicas de Ubuntu, pero es utilizado como guía para el proceso de desarrollo de Nova. Debian posee un servidor de Jenkins (integración continua) donde automatiza sus procesos, entre ellos el de construcción de imágenes, pero al utilizar como instalador Debian-installer los puntos de contacto entre estos pasos y los utilizados en Nova Escritorio son casi nulos por lo que no se tiene en cuenta para la presente investigación (Debian, 2018).

1.4.1 Imagen de disco

Una imagen de disco (imagen en lo adelante) es una réplica exacta, sector por sector, de un disco, CD, DVD u otro dispositivo de almacenamiento. Las imágenes de disco se guardan en un único archivo que puede comprimirse y almacenarse en un disco duro, unidad USB, CD o DVD. Dependiendo de los parámetros que se elija, las imágenes pueden guardarse con las extensiones de archivo .img, .cdr, .dmg y .iso entre otras (ISO, 1988).

Las imágenes pueden utilizarse para restaurar completamente un sistema después de que se haya bloqueado, de un fallo o de un ataque de virus. También son útiles para instalar múltiples estaciones de trabajo o equipos con una configuración exacta. En lugar de configurar cada equipo, uno por uno, se puede establecer un sistema con las configuraciones deseadas, crear una imagen de disco e instalarla en el resto de los equipos (ISO, 1988).

Una imagen es un espejo de un dispositivo (partición). Esta copia contiene toda la información presente en el sistema original. Una característica especial de una imagen es que el formato del archivo está especialmente comprimido y toma mucho menos espacio en disco que el original. En caso de que los datos originales de una unidad se pierdan o dañen, puede ser restaurados sin ninguna dificultad (O&O Software, 2017).

[8] Metapaquete: permiten agrupar varios programas junto a sus dependencias para su instalación de una vez.

En la presente investigación se adopta la extensión .iso para la construcción de las imágenes de discos, pues es la utilizada por las distribuciones de GNU/Linux Ubuntu a la cual Nova se circunscribe. Por ellos se puede definir que un archivo ISO es un archivo de imagen de disco.

1.4.2 Pasos para la construcción de una imagen

Cuando se analizan los pasos para la construcción de una imagen, el mismo se puede dividir en varios momentos (Beekmans, 2010):

1. Construcción de sistema base.
2. Instalación de aplicaciones en el sistema base.
3. Construcción de estructura de carpeta de la imagen y su contenido.
4. Construcción de la imagen.

La distribución cubana de GNU/Linux Nova utiliza como base a Ubuntu. De Ubuntu hereda su proceso de desarrollo, su sistema de empaquetado y su estructura. En el caso de Nova Escritorio se basa en los procesos definidos para Ubuntu Desktop. Nova Ligero y Nova Servidores heredan su proceso de desarrollo de Lubuntu, versión ligera de Ubuntu, y Ubuntu Server respectivamente.

Ubuntu es un sistema operativo de código abierto que funciona correctamente en computadores personales, tecnologías en la nube y todos los dispositivos que se conectan a internet (Canonical, 2018). Precisamente esta fue una de las causas que conllevó a su utilización como base de Nova, así como su estabilidad y popularidad dentro del mundo del software libre. Debido a esto, en la presente investigación, se analizará cada uno de los pasos que se realizan para la construcción de Ubuntu y que coinciden con el desarrollo de Nova.

1.4.3 Sistema base

Un sistema base es aquel que cuenta con las aplicaciones necesarias e imprescindibles para el funcionamiento del sistema (Beekmans, 2010).

Un sistema base es aquel que sirve de base para cualquier otro. En esta base es donde se realizan los cambios que constituyen distintivos de una personalización (O&O Software, 2017).

Las aplicaciones incluidas en un sistema base de Ubuntu y Nova por lo general son: Bash, Check, Coreutils, E2fsprogs, File, Flex, Gawk, GCC, Gettext. Glibc. GMP. Grep, IPRoute, Kdb, Kmod, Libpipeline, Linux, Man-DB, Man-pages, Perl, Psmisc, Readline, Shadow, Tar, TCL y Util-Linux (Beekmans, 2010). Aunque se pueden incluir otras para enriquecer el sistema. Todas estas aplicaciones están etiquetadas de “important” en el repositorio de código fuente y binarios de las distribuciones analizadas.

1.4.4 Estructura del sistema base

A continuación, se describen cuáles son los directorios que componen la base y qué contiene cada uno de ellos.

Tabla 1: Estructura de carpetas de un sistema base (José & González, 2009).

/ Directorio raíz	/bin	Contiene programas ejecutables básicos para el sistema.
	/boot	Contiene los ficheros necesarios para el arranque del sistema.
	/dev	Contiene los ficheros correspondientes a los dispositivos: sonido, impresora, disco duro, lector de cd/dvd, video, etc.
	/etc	Contiene ficheros y directorio de configuración.
	/home	Contiene los directorios de trabajo de los usuarios. Cada usuario tiene su propio directorio en el sistema dentro de /home/.
	/lib	Contiene las librerías compartidas y los módulos del núcleo.
	/media	Dentro de este directorio se montan los dispositivos como el CD-ROM, memoria USB, dispositivos portables, etc.
	/opt	Directorio reservado para instalar aplicaciones.
	/sbin	Contiene los ficheros binarios ejecutables del sistema operativo.
	/srv	Contiene los datos de los servicios proporcionados por el sistema.
	/tmp	Directorio de archivos temporales.
	/usr	Aquí se encuentran la mayoría de los archivos del sistema, aplicaciones, librerías manuales, juegos, etc. Es un espacio compartido para todos los usuarios.
	/var	Contiene archivos administrativos y datos que cambian con frecuencia: registro de errores, bases de datos, colas de impresión, etc.
	/root	Directorio de trabajo del administrador del sistema (root).
	/proc	Aquí se almacenan datos del núcleo e información sobre procesos.

Sobre esta base se instalan aplicaciones, tales como el entorno de escritorio, navegador web, navegador de archivos, así como todas las que sean definidas en el proceso de ingeniería de requisitos. A la hora de realizar personalizaciones específicas para alguna entidad se incluyen las necesarias para el correcto funcionamiento del proceso que se desee informatizar y depende del cliente la correcta definición de las mismas.

1.4.5 Estructura de carpetas de la imagen y su contenido

La estructura de carpetas de una imagen de Nova Escritorio es similar a la de una imagen de Ubuntu

Desktop, teniendo en cuenta que ambos utilizan las aplicaciones Ubiquity como instalador y casper para el liveCD[9]. Se diferencian al caso de Lubuntu pues este último utiliza Debian-Installer[10] que requiere de otro conjunto de características para realizar el proceso de instalación, esta estructura es similar a la de Nova Ligero ya que constituye la base para su construcción. A continuación, se muestra la estructura de carpetas de Ubuntu Desktop y por tanto la de Nova Escritorio.

Una imagen ISO de Ubuntu Desktop está compuesta por la siguiente información:

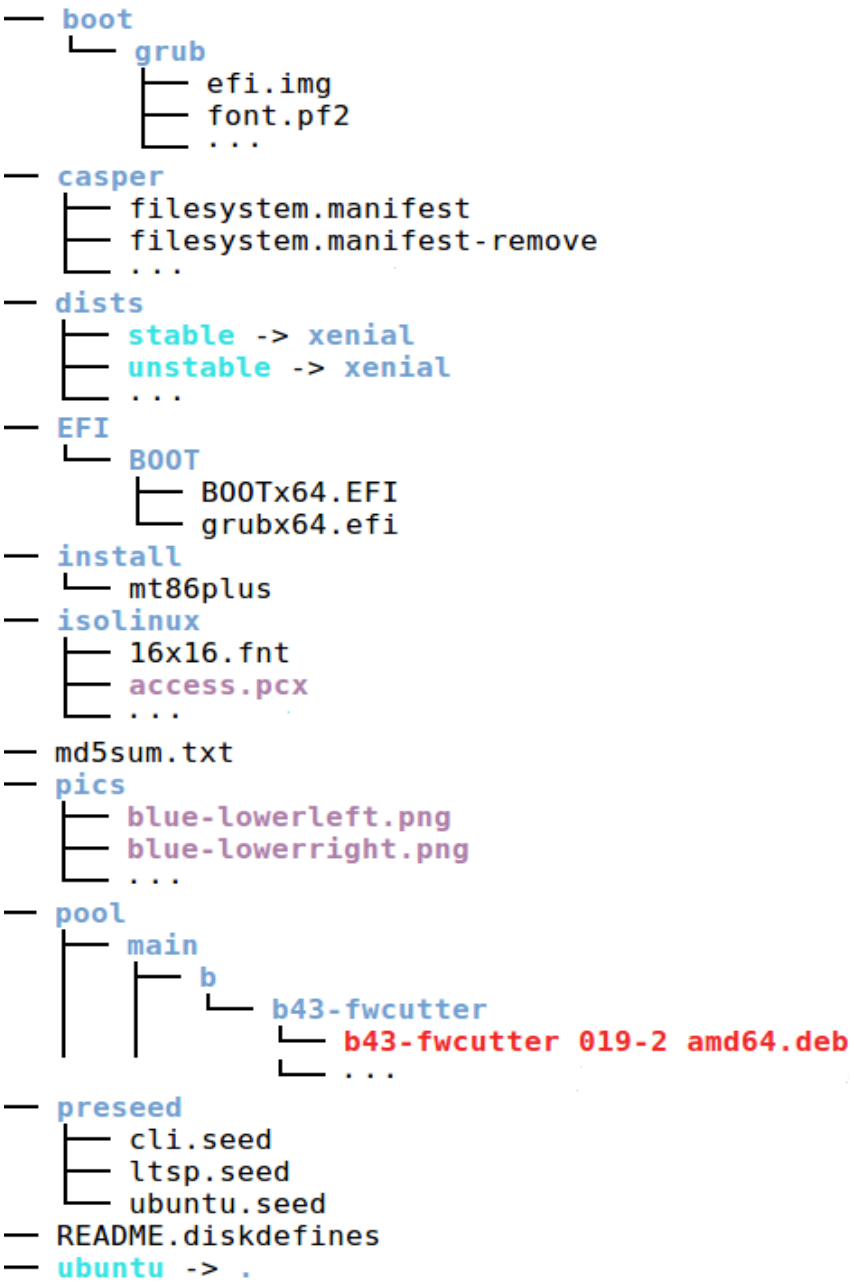

Figura 1: Estructura de una imagen (Fuente: elaboración propia).

[9] liveCD: es un completo sistema operativo Linux en un CD (Petersen & Henrichsmeyer, 2008).
[10] Debian-Installer: es el programa de instalación para Debian GNU/Linux.

- boot: almacena las configuraciones del cargador de arranque del sistema para instalar en particiones GPT[11]. Esto incluye las características gráficas que el mismo debe poseer, así como la dirección exacta del núcleo y del archivo de configuración del liveCD dentro del sistema de fichero.
- casper: contiene los archivos que utiliza el sistema para cargar el liveCD. Esto incluye el sistema comprimido (.squashfs), los archivos que necesita el instalador para su funcionamiento (.manifest, .manifest-desktop), el tamaño del sistema comprimido una vez que se descomprime (.size), el sistema de inicio de Linux (initrd.lz) y el núcleo del sistema (vmlinuz).
- dists: almacena los índices del repositorio interno, esto es utilizado por el instalador si necesita instalar alguna nueva aplicación.
- EFI: contiene las configuraciones necesarias para cargar el sistema con EFI.
- isolinux: almacena las configuraciones del cargador de arranque para instalar en particiones MBR[12]. Esto incluye las características gráficas que el mismo debe poseer, así como la dirección exacta del núcleo y del archivo de configuración del liveCD dentro del sistema de fichero.
- pool: contiene los paquetes de código binario del repositorio interno, esto es utilizado por el instalador si necesita instalar alguna nueva aplicación.
- .disk: incluye informaciones del sistema operativo, tales como nombre, versión, arquitectura, etc.
- md5sum.txt: es un archivo que almacena las sumas de verificación de cada uno de los archivos de la imagen.
- preseed: conjuntos de instrucciones propias del sistema que permite habilitar y deshabilitar funcionalidades específicas.
- ubuntu: enlace simbólico a la misma carpeta donde se encuentra la estructura de la imagen.
- pics: conjunto de imágenes heredadas de Debian.

En el caso específico de Nova Escritorio lo único que varía es el repositorio de aplicaciones que es modificado con las características de Nova y el enlace simbólico que no se nombra ubuntu, sino nova.

1.5 Pruebas definidas para Nova

A lo largo de todo el proceso de construcción de la personalización de GNU/Linux Nova se le aplican pruebas al producto para verificar su calidad. A continuación, se describen las más importantes:

---

[11] GPT: es un estándar para la colocación de la tabla de particiones en un disco duro físico (Nikkel, 2009).
[12] MBR: es el primer sector de un dispositivo de almacenamiento de datos, como un disco duro (Al Sadi, 2016).

1- Pruebas unitarias: enfoca los esfuerzos en la unidad más pequeña del diseño de software: el componente o módulo de software. Al usar la descripción del diseño de componente como guía, las rutas de control importante se prueban para descubrir errores dentro de la frontera del módulo (Pressman, 2013). Si extrapolamos esto al desarrollo de Nova, se podría definir como módulo cada una de las aplicaciones que se integran al entorno de escritorio. El desarrollador responsable de su creación o modificación debe probarla en el sistema y verificar su correcto funcionamiento.
2- Pruebas de integración: aunque una aplicación funcione por sí sola, nada garantiza que funcione junto con el resto de las aplicaciones que componen el ecosistema de software. Los datos pueden perderse a través de una interfaz; un componente puede tener un inadvertido efecto adverso sobre otro, etc. (Pressman, 2013). En el caso de Nova las aplicaciones se compilan y empaquetan, se suben a un repositorio de pruebas y se instalan en alguna imagen para verificar errores, tanto de la propia aplicación una vez instalada o de dependencias incumplidas, así como posibles conflictos con otras aplicaciones
3- Pruebas internas: cuando se tiene alguna versión estable del sistema operativo se le realizan pruebas exploratorias en busca de no conformidades que no fueron detectadas previamente.
4- Pruebas de aceptación: cuando se construye un software a la medida para un cliente, se realiza una serie de pruebas de aceptación a fin de permitir al cliente validar todos los requerimientos. Realizada por el usuario final en lugar de por los ingenieros de software puede durar de semanas a meses, y mediante ella descubrir errores acumulados que con el tiempo puedan degradar el sistema (Pressman, 2013).
5- Pruebas de seguridad: intenta verificar que los mecanismos de protección que se construyen en un sistema en realidad los protegerán de cualquier penetración impropia (Pressman, 2013). En el caso de Nova los encargados de realizar estas pruebas es la Dirección de Seguridad Informática que verifica que las aplicaciones utilizadas en el sistema no presentan vulnerabilidades que puedan poner en peligro información sensible y que las herramientas que permiten cumplir con las políticas de seguridad establecidas en la Universidad son completamente funcionales en el nuevo sistema.

1.5.1 Deficiencias detectadas en las pruebas

A pesar de la existencia de pruebas a lo largo de todo el ciclo de vida del proyecto, las mismas son realizadas fundamentalmente a la correcta documentación de la ingeniería de software y al producto en sus diferentes momentos de desarrollo. Pero no existen pruebas dentro del proceso de desarrollo que posibilite la detección temprana de errores. Tampoco están definidas las acciones correctivas que se deben tomar ante una problemática, ni quiénes son los responsables de darle solución al problema.

Los principales errores que inciden en el proceso de construcción de una imagen son debido a la

inexistencia de alguna aplicación en el repositorio o a errores de dependencias. Resolver estos errores resulta muy complicado, pues rastrear una dependencia incumplida es extremadamente tedioso y la herramienta que utiliza Nova para la instalación de sus paquetes (APT) no se encuentra configurada para facilitar este trabajo. También es común que se quede información innecesaria dentro de la imagen, tal es el caso de instalables que son descargados por APT y de los logs que son creados por las aplicaciones instaladas.

Las aplicaciones que componen una imagen no son las mismas antes y después de la instalación. Tal es el caso del instalador que se necesita esté presente antes de la instalación, no siendo así luego de la misma. Estas diferencias se especifican en dos archivos presentes en la imagen, pero es muy común que queden dependencias instaladas y que no son necesarias luego de desinstalar este tipo de aplicaciones. Ubuntu, en su proceso de construcción de imágenes especifica qué aplicaciones deben ser desinstaladas teniendo en cuenta que ubiquity y casper son las únicas herramientas que, en su proceso, deben estar antes pero no después de la instalación. En la versión 6.0 de Nova se decidió incluir una nueva aplicación, Gparted, que estuviera presente en el LiveCD pero no una vez instalado el sistema en el ordenador, esto provocó un cúmulo de aplicaciones innecesarias presentes en el sistema instalado.

Actualmente no existe alguna prueba que detecte estas insuficiencias ni un procedimiento definido para solucionarlas.

1.6 Procesos que intervienen en la construcción de Nova

- Construcción del repositorio de código fuente: se crea un repositorio que contiene el código fuente de las aplicaciones etiquetadas de “important” en el repositorio oficial de Ubuntu. Estas son las aplicaciones básicas que necesita un sistema para funcionar correctamente.
- Modificación del código fuente de algunas aplicaciones, compilación y empaquetado: se modifica el código fuente de varias aplicaciones que permiten darle identidad a la distribución cubana de GNU/Linux Nova. Entre ella se encuentran: distro-info y distro-info-data (información de la distribución), devscripts (scripts para el uso de los desarrolladores), nova-keyring (llaves públicas de Nova), debootstrapt (construcción de sistemas base). Luego se compilan y empaquetan todas las aplicaciones albergadas en el repositorio para crear los instalables de las mismas.
- Construcción del repositorio de código binario: con los instalables generados se crea el repositorio de código binario que los almacena. En este punto se puede decir que se posee un repositorio de código fuente y binarios de los paquetes básicos que necesita un sistema para funcionar correctamente.
- Modificación del entorno de escritorio: se incluye el código fuente del entorno de escritorio y todas sus dependencias en el repositorio y se comienza a modificar teniendo en cuenta la arquitectura de información definida en el departamento de Sistemas Operativos.

- Modificación de aplicaciones: se incluye el código fuente de varias aplicaciones que responden a los requisitos detectados en la ingeniería de requisitos, así como sus dependencias, y se comienzan a modificar, siempre que sea necesario, para que se integren correctamente con el entorno de escritorio.
- Construcción de imágenes de Nova: a lo largo de cada uno de los procesos anteriores se construyen imágenes para verificar el correcto funcionamiento de cada uno de ellos y para detectar posibles incompatibilidades entre algunas de las aplicaciones requeridas.

Se puede concluir que los proceso que están más relacionado con la construcción de las imágenes son los que inciden en el repositorio de código fuente y binarios, si se tiene en cuenta que para la construcción de una imagen la entrada es precisamente los paquetes que componen el repositorio, los cuales pasan por un conjunto de pasos para ser convertidos en ella. Por tanto, se puede determinar que ante cualquier cambio en el repositorio se debe actualizar la imagen de forma tal que la misma posea las últimas modificaciones realizadas, pero ¿cómo se podría saber con exactitud cuándo hay un cambio en el repositorio?

La utilización de una herramienta de integración continua permitiría automatizar tareas que una vez finalizadas con éxito dispararía de forma automática la tarea de construcción de la imagen. En el departamento de Sistemas Operativos se maneja la idea de la utilización de una herramienta de integración continua para la automatización de los procesos relacionados con el repositorio de código fuente, esto permitiría aumentar la integración del repositorio de código fuente y binarios con el proceso de construcción de imágenes permitiendo eliminar uno de los principales problemas que afectan al producto final: en muchas ocasiones, aunque se soluciona una no conformidad presente en un paquete, el producto final (la imagen) sigue arrastrando dicho problema debido a la falta de comunicación entre los mantenedores de paquetes, el administrador de la configuración (responsable del repositorio) y los desarrolladores que se encargan de la actualización de las imágenes. También podría programarse alguna tarea dentro del sistema operativo que compruebe la existencia de actualizaciones en el repositorio (esta información puede ser detectada con el uso de APT) en caso afirmativo ejecutar un conjunto de scripts que automaticen el proceso.

Teniendo en cuenta esto se puede concluir que el procedimiento que se propone debe ser implementable a través de lenguajes script y poseer pruebas asociadas que permitan detectar errores de forma temprana, posibilitando su inclusión en la herramienta de integración continua utilizada en el departamento de Sistemas Operativos.

1.5 Proceso de migración en Cuba

El proceso de migración a software libre y código abierto en Cuba se realiza con el objetivo de lograr un desarrollo autóctono en el campo de la informática, teniendo total capacidad de decisión sobre las tecnologías y en la forma en que se desarrollan y usan las mismas (Pérez et al., 2015). El mismo

nace a partir del acuerdo 084/2004 del consejo de ministros.

La migración que lleva a cabo el país a estándares de software libre y código abierto, dentro de la cual Nova juega un papel fundamental, se ha visto afectada por la manera dispersa en que se confeccionan las imágenes. En entrevista realizada a especialistas en migración (ver Anexo 3) se detectó que la amplia gama de estaciones de trabajo que conviven en el entorno tecnológico cubano provoca que a la hora de ir a una institución a migrar es necesario llevar tres dispositivos externos con las tres versiones principales de la distribución: Nova Escritorio (i386 y amd64) y Nova Ligero. En dependencia de las características de la estación de trabajo se utiliza una de ellas. Los dispositivos externos utilizados son normalmente memorias flash de como mínimo 8Gb de capacidad, a pesar de ello su capacidad de almacenamiento se desaprovecha pues una imagen suele ocupar alrededor de 1,5 Gb y el resto del dispositivo queda vacío.

Se determina que la posibilidad de reunir en un mismo booteable las tres distribuciones sería provechoso pues permitiría la utilización de menos dispositivos externos para realizar la tarea, sería más humano para el especialista de migración que tendría en un mismo dispositivo todas las soluciones y aprovecharía mejor las capacidades de almacenamiento de los dispositivos.

Teniendo en cuenta esto, se concluye que Nova Ligero es una entrada del presente procedimiento.

Conclusiones del capítulo

- El análisis del sistema posibilitó determinar la incompatibilidad del mismo con UEFI, determinando que este problema viene dado por la falta de varios parámetros en el comando utilizado para la construcción de las imágenes y a que Nova no posee un certificado firmado por Microsoft UEFI CA para su grub.
- La imposibilidad de crear imágenes de discos desde la herramienta definida en las Especificación de Requisitos de Nova y luego de aplicar la metodología de prueba y error, posibilitó determinar la incompatibilidad de la herramienta usada en Nova para la construcción de las imágenes y que su sustitución por xorriso soluciona el problema.
- El análisis del proceso de construcción de imágenes, así como el estudio de los principales conceptos asociados al problema planteado posibilitó sentar las bases para el desarrollo de la investigación y conocer las características del objeto de estudio.
- La descripción de las pruebas que se le aplican al sistema posibilitó detectar que no existen pruebas definidas dentro de dicho proceso que permitan identificar dependencias incumplidas en la instalación de aplicaciones, así como información y aplicaciones innecesarias.
- El estudio de los procesos que intervienen en la construcción de las imágenes posibilitó detectar la necesidad de una mayor integración entre el repositorio de código fuente y el procedimiento de construcción de imágenes.

- El análisis de la estructura de las imágenes, en conjunto con la opinión de los especialistas de migración posibilitó comprender que, teniendo en cuenta las condiciones de nuestro país, resulta complejo tener las personalizaciones por separado y resultaría provechoso que las más importantes estén integradas en una sola imagen.

CAPÍTULO 2. ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA PROPUESTA DE SOLUCIÓN

En el presente capítulo, se propone una nueva estructura para las imágenes de la distribución de GNU/Linux Nova teniendo en cuenta los elementos detectados en el capítulo anterior. Se define el procedimiento de construcción de imágenes, pruebas que permiten identificar problemas que anteriormente no se detectaban en etapas tempranas del desarrollo así como los elementos arquitectónicos y de implementación de la solución.

2.1 Nueva estructura de carpetas

Teniendo en cuenta la necesidad de que en una sola imagen convivan las distribuciones más importantes de Nova, es necesario hacerle una modificación a su estructura teniendo en cuenta que la misma debe estar compuesta por la unión entre cada una de ellas. Se puede considerar a cada imagen como un conjunto, en el presente caso Nova Escritorio i386 (C1), Nova Escritorio amd64 (C2) y Nova Ligero (C3) y la nueva imagen (P), teniendo en cuenta los fundamentos de la teoría de conjuntos. Se puede apreciar que, debido a la necesidad de tener a Nova Ligero integrada con Nova Escritorio, esta distribución es una entrada del presente procedimiento.

$$P = C1 \cup C2 \cup C3$$

Los principales cambios consisten en la aparición de una carpeta de nombre “live”, propia de Nova Ligero que es donde se almacena el sistema base para esa distribución. El repositorio interno de la imagen aumenta considerablemente si se tiene en cuenta que Nova Ligero instala la mayoría de las aplicaciones a partir del repositorio con la utilización de Debian-installer. Pero, ¿cómo unificar las carpetas “casper” de Nova Escritorio i386 y amd64 cuando las mismas comparten la misma información y con los mismos nombres de archivos, pero para arquitecturas diferentes?

Teniendo en cuenta que ambas personalizaciones deben poseer los mismos paquetes instalados se puede determinar que pueden compartir los mismos archivos “filesystem.manifest” y “filesystem.manifest-desktop”. Se propone separar los elementos específicos de cada distribución en carpetas dentro de la carpeta “casper” con el nombre de la arquitectura que los mismos representan. Por lo que el sistema comprimido (“filesystem.squashfs”), el núcleo (“vmlinuz”) y el cargador de arranque (“initrd.lz”) de Nova Escritorio i386 estarán dentro de una carpeta llamada “i386”, los de amd64 dentro de una carpeta “amd64” y así sucesivamente para las arquitecturas que se vayan incorporando a la distribución cubana de GNU/Linux.

Hasta este punto aún no funciona correctamente la propuesta de solución debido a que casper no es capaz de entender dónde tiene que buscar la información correspondiente a cada una de las distribuciones. Es por ello que se hace necesario modificar este paquete. Es preciso obtener la arquitectura del sistema en el que se instala casper y pasársela al archivo de configuración

“casper.conf”, para ello se agregan el Makefile[13] las siguientes líneas:

```
IMAGE_ARCH := $(Shell dpkg –print-architecture)

all:

  sed –i –e ‘s/\(IMAGE_ARCH=”\).*”/\1’$(IMAGE_ARCH)”’/g’ casper.conf
```

En el archivo script/casper modificar el directorio donde se busca la información para incluirle la arquitectura.

```
LIVE_MEDIA_PATH = casper/$IMAGE_ARCH
```

2.2 Instalación con EFI

La imagen debe ser construida usando los parámetros requeridos para una correcta instalación en discos duros con tabla de particiones GPT, lo cual quedará definido más adelante, pero también es necesario modificar el paquete grub-common que se encarga de generar la información de la distribución en cuestión que es procesada por el firmware UEFI para determinar si es un sistema válido, o sea certificado. Al ser un paquete de Ubuntu, no incluye a Nova como una de sus distribuciones, por lo que es necesario incluirla. Para ello se modifica el archivo “grub-install.c” y se le agregan las siguientes líneas:

```
If (strcmp (efi_distributor, “nova”) == 0)

 efi_distributor = “ubuntu”;
```

Esto permite que el sistema sea camuflado como Ubuntu y se procede a empaquetar nuevamente esta aplicación. La misma genera varios paquetes que se encuentran en el repositorio interno de la imagen y en el repositorio de aplicaciones de Nova y que deben ser sustituidos por los nuevos. Estos son:

- grub-efi
- grub-efi-amd64
- grub-efi-amd64-bin

Una vez concluida esta sustitución ya el sistema permitirá la instalación en particiones GPT.

2.3 Procedimiento de construcción de imágenes de Nova Escritorio

[13] Makefile: Es un archivo que contiene un conjunto de directivas utilizadas por una herramienta de automatización de compilación para generar un objetivo / meta (IEEE, 2018).

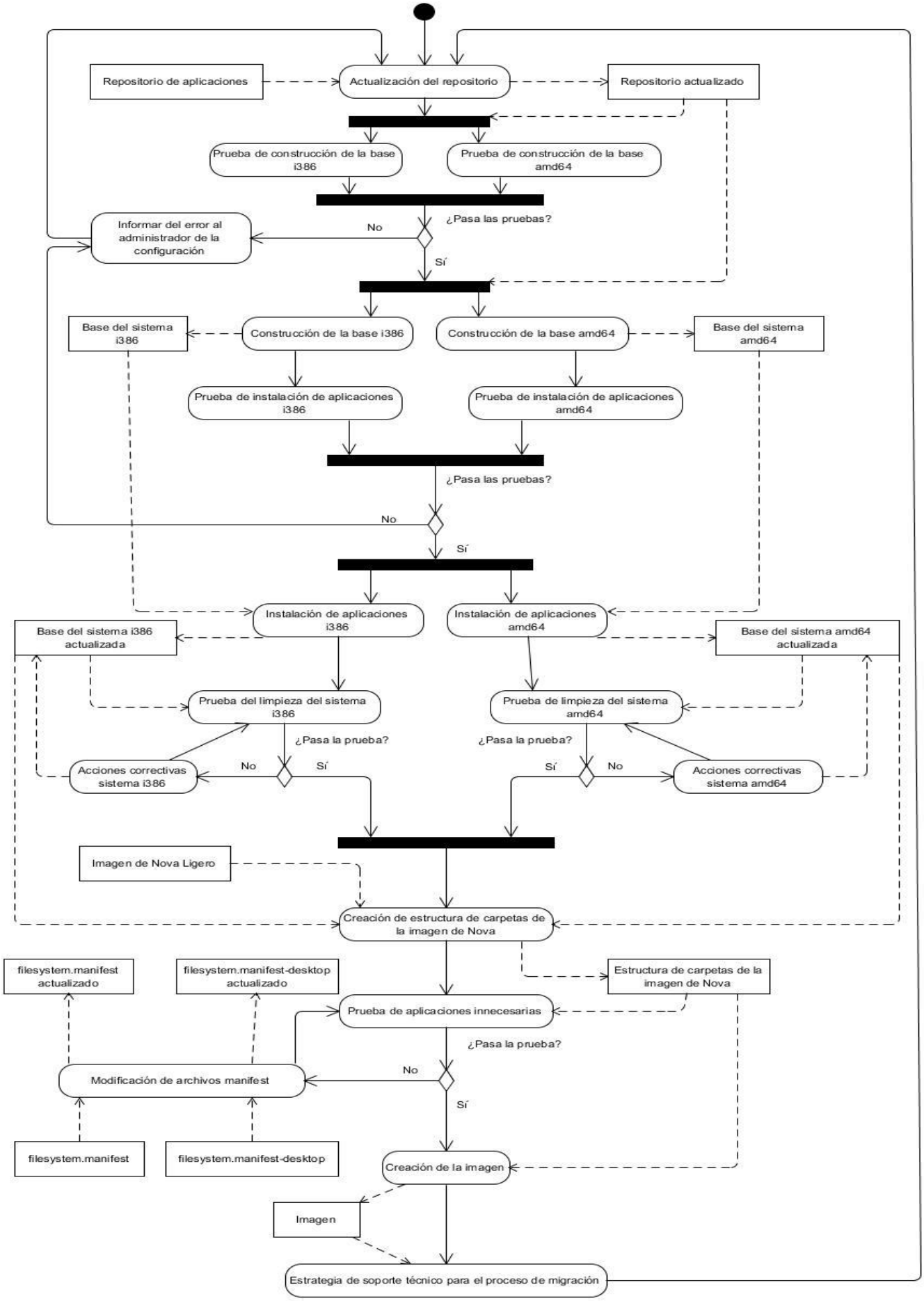
Repositorio de aplicaciones
Actualización del repositorio
Repositorio actualizado
Prueba de construcción de la base i386
Prueba de construcción de la base amd64
¿Pasa las pruebas?
No
Informar del error al administrador de la configuración
Sí
Base del sistema i386
Construcción de la base i386
Construcción de la base amd64
Base del sistema amd64
Prueba de instalación de aplicaciones i386
Prueba de instalación de aplicaciones amd64
¿Pasa las pruebas?
No
Sí
Instalación de aplicaciones i386
Instalación de aplicaciones amd64
Base del sistema i386 actualizada
Base del sistema amd64 actualizada
Prueba del limpieza del sistema i386
Prueba de limpieza del sistema amd64
¿Pasa la prueba?
¿Pasa la prueba?
Acciones correctivas sistema i386
No
Sí
Sí
No
Acciones correctivas sistema amd64
Imagen de Nova Ligero
Creación de estructura de carpetas de la imagen de Nova
filesystem.manifest actualizado
filesystem.manifest-desktop actualizado
Estructura de carpetas de la imagen de Nova
Prueba de aplicaciones innecesarias
¿Pasa la prueba?
Modificación de archivos manifest
No
Sí
filesystem.manifest
filesystem.manifest-desktop
Creación de la imagen
Imagen
Estrategia de soporte técnico para el proceso de migración

Figura 2: Procedimiento de construcción de imágenes de Nova.

2.3.1 Actualización del repositorio

El procedimiento comienza una vez que se realice algún cambio en el repositorio de código fuente y binarios, que puede ser la actualización, modificación o incorporación de algún paquete en el mismo. La entrada de este paso es el repositorio de aplicaciones y la salida el repositorio actualizado. El responsable de realizar esta tarea es el administrador de la configuración. Precisamente en este paso se incluye la modificación de los paquetes antes mencionados en la investigación con los elementos definidos: casper y grub-common.

2.3.2 Prueba de construcción de la base i386 / amd64

Para la construcción de la base se pueden utilizar dos variantes fundamentales, hacerlo desde cero con los procedimientos descritos en Linux from Scratch (Linux desde cero) o utilizando la aplicación debootstrap que ya se encuentra incluida en los repositorios de Ubuntu y que automatiza el proceso. Teniendo en cuenta esto se utilizará esta última.

Esta prueba consiste en realizar el proceso de construcción de la base en una carpeta temporal.

```
export RUTA=/home/

CHROOT_TARGET=$(mktemp –d –p $RUTA chroot-installation-2017.XXXXXXXXXX)

debootstrap –arch=amd64 –component=principal,extendido 2017 $CHARROT_TARGET
```

La arquitectura, los componentes y la distribución varían en dependencia de la personalización que se esté realizando. Si este comando falla, se podrá visualizar qué aplicación fue la causante del problema. La causa del problema es debido a errores en el repositorio de código fuente y binarios ya que la aplicación detectada es inexistente o posee algún problema con la suma de verificación que impide su descarga, por tanto, lo correcto sería avisarle automáticamente al administrador de la configuración para que solucione el problema.

La entrada de esta tarea es el repositorio de aplicaciones actualizado y se realiza de forma paralela tanto para la arquitectura i386 como para la arquitectura amd64. El encargado de realizar esta prueba es el desarrollador.

2.3.3 Construcción de la base i386 / amd64

Constituyen dos tareas que se realizan de forma paralelas. Se deben poseer dos carpetas donde se realizará todo el proceso para i386 y para amd64. En el siguiente ejemplo el nombre definido para las carpetas será “chrooti386” y “chrootamd64”.

1. Se hace uso de la aplicación “debootstrap”. La sintaxis sería la siguiente: *debootstrap --arch=amd64 --components=principal,extendido 2017 chrootamd64 http://novarepo.uci.cu/nova* para la base amd64 y *debootstrap --arch=i386 --*

components=principal,extendido 2017 chrooti386 http://novarepo.uci.cu/nova para la base i386*,* donde --arch constituye la arquitectura del sistema que se desea construir, --components las ramas del repositorio que se van a utilizar, 2017 la distribución de la personalización, chrootamd64 y chrooti386 las carpetas que se utilizarán para alojar las bases y por último el repositorio en cuestión de donde se descargarán las aplicaciones necesarias para dicha construcción.

La entrada de esta tarea constituye el repositorio de aplicaciones y la salida las bases de ambas arquitecturas presentes actualmente en Nova. El responsable es el desarrollador.

2.3.4 Prueba de instalación de aplicaciones i386 / amd64

Se procede a configurar la celda tal cual se define en Instalación de aplicaciones i386/amd64 que se explica en el próximo punto, aunque se le incluyen un conjunto de modificaciones. El principal problema para la presente actividad es cuando ocurre un error de dependencias, ya que la herramienta de administración de aplicaciones de Nova solo brinda información referente a la aplicación que se trató de instalar y tuvo el error, pero no muestra cuál fue la dependencia que lo originó, esto desencadena una búsqueda exhaustiva y tediosa, pues una aplicación depende de muchas otras, que a la vez dependen de otras tantas y así de forma recursiva.

APT posee un gran conjunto de parámetros para mostrar más o menos información. El estudio realizado permitió determinar que una correcta configuración de la herramienta permite visualizar la información requerida detectando lo siguiente:

Debug::pkgDepCache::Marker: genera mensaje de debug describiendo qué paquetes se marcan como retenidos/instalados/eliminados mientras el solucionador de problemas (ProblemResolver) de APT hace su trabajo. Cada adición o eliminación puede disparar acciones adicionales; se muestran identados por dos espacios dentro de la entrada original. El formato para cada entrada es MarkInstall seguido por <nombre_del_paquete> <a.b.c -> d.e.f | x.y.z> (sección) donde a.b.c es la versión actual del paquete, d.e.f es la versión considerada para la instalación y x.y.z es la versión más nueva presente en el repositorio, pero no considerada para la instalación. Los últimos 2 pueden ser omitidos si no hay o si es el mismo de la versión instalada.

Debug::pkgDepCache::AutoInstall: genera mensajes de debug describiendo qué paquetes serán instalados automáticamente para resolver dependencias.

Debug::pkgProblemResolver: tracea la ejecución del solucionador de dependencias. Esto aplica solo cuando un complejo problema de dependencia es encontrado.

Debug::pkgPackageManager: muestra el mensaje de estado de tracear los pasos de ejecución

cuando se invoca DPKG[14].

Teniendo en cuenta esto, se puede crear un nuevo archivo de configuración de APT y activar los parámetros anteriormente descritos:

cat > /etc/apt/apt.conf.d/80debug << APTEOF

Debug::pkgDepCache::Marker "true";

Debug::pkgDepCache::AutoInstall "true";

Debug::pkgProblemResolver "true";

Debug::pkgPackageManager "true";

APTEOF

A partir de este punto APT mostrará de forma detallada cada uno de los pasos que ejecuta a la hora de instalar una aplicación y en caso de un fallo de dependencia se conocería con exactitud qué dependencia fue la que falló. Esto significa que esta dependencia tiene problemas en el repositorio de código fuente y binarios, bien porque no existe o porque tiene algún error en la suma de verificación que imposibilita su descarga. Por lo que el administrador de la configuración es el encargado de solucionar esta problemática en el repositorio de aplicaciones.

La entrada de esta prueba son las bases anteriormente creadas y el responsable de realizarla es el desarrollador.

2.3.5 Instalación de aplicaciones i386 / amd64

En este punto ya se tiene el sistema base para amd64 e i386 y se procede a instalar las aplicaciones necesarias de la personalización en cada uno de ellos.

1. Montar la información referente a los controladores y dispositivos conectados al ordenador en el sistema base creado *mount --bind /dev chrootamd64/dev* y *mount --bind /dev chrooti386/dev*.
2. Crear un entorno virtual donde la raíz del sistema pasará a ser la carpeta "chrooti386" y "chrootamd64" donde se tienen las bases construidas. *chroot chrooti386/* y *chroot chrootamd64/*. Todas las acciones que se realicen a partir de este momento no se realizarán en la estación de trabajo en sí, sino en el sistema de ficheros creado en las carpetas "chrooti386" y "chrootamd64".
3. A partir de este paso, tanto para la base de amd64 como para la de i386 el procedimiento será exactamente igual. Se montan los sistemas de ficheros procfs, sysfs y devpts que se utilizan para permitir el acceso a la información del núcleo sobre los procesos y para exportar

[14] DPKG: base del sistema de gestión de paquetes Debian en un sistema.

información presente en el árbol de los dispositivos. *mount none –t proc proc mount none –t sysfs sys mount none –t devpts dev/pts.*

4. Exportar la variable del sistema LC_ALL para usar el lenguaje por defecto de la base creada. *export LC_ALL=C.*
5. Actualizar el repositorio *apt-get update.*
6. Instalar las aplicaciones que formarán parte de la personalización, apt-get install linux-image-generic (núcleo), *apt-get install nova-base* (aplicaciones básicas de Nova), *apt-get install nova-escritorio* (entorno de escritorio y sus aplicaciones por defecto), *apt-get install ubiquity-frontend-gtk* (instalador), *apt-get aplicaciones* (otras aplicaciones que se deseen incluir), *apt-get casper* (crear sistema liveCD). Las aplicaciones que se instalan dentro del sistema base varían en dependencia de los requerimientos del usuario, es por ellos que las aplicaciones instaladas en el ejemplo no tienen que ser necesariamente las que se utilicen para una personalización específica.
7. Limpiar el historial tanto de las aplicaciones instaladas como de los comandos ejecutados *apt-get clean, history –c.*
8. Salir del entorno virtual y desmontar los sistemas de ficheros previamente montados *exit, umount –lf chrooti386/proc, umount –lf chrooti386/sys, umount –lf chrooti386/dev, umount –lf chrootamd64/proc, umount –lf chrootamd64/sys, umount –lf chrootamd64/dev.*

La entrada de este procedimiento son las bases de las dos arquitecturas presentes en Nova y la salida las base actualizadas con las aplicaciones instaladas. El responsable es el desarrollador.

2.3.6 Prueba de limpieza del sistema i386 / amd64

Un sistema no debe contener información innecesaria. Un ejemplo de ello es la presencia de aplicaciones descargadas por APT. Cuando se va a instalar una aplicación el procedimiento es el siguiente: APT busca la aplicación en los índices del repositorio, así como sus dependencias, de las dependencias busca también sus dependencias y así sucesivamente. Todos estos paquetes son descargados en el directorio /var/cache/apt/archives/ para luego ser instalados con el uso de la herramienta DPKG. Estos archivos normalmente ocupan espacio innecesario y no deberían estar presentes en el sistema una vez terminado el procedimiento.

También es posible instalar aplicaciones a partir de un .deb que se puede almacenar en cualquier lugar del sistema. Por otra parte, antes de comprimir el sistema no debe quedar ninguna información en /tmp/ ni en /var/log/. En este paso se propone hacer una búsqueda de este tipo de archivos que no deben estar presentes en el sistema y que lamentablemente se quedan presenten en la gran mayoría de las imágenes que se realizan. En caso de encontrarlos se puede proceder automáticamente a su eliminación.

*logs = $(find / -name *.log)*

*debs = $(find / -name *.deb)*

*ddebs = $(find / -name *.ddeb)*

for i in $logs; do rm –R $i; done

Esto se puede hacer para cada una de las variables definidas y para tantos formatos como se desee e.j. tar, tar.gz, tar.xz, dsc, etc. que son formatos propios de los paquetes de código fuente y no tienen por qué estar presentes en el sistema. Se propone que cualquier aplicación o información que se desee copiar para responder a algún requisito específico se almacenen en el directorio /tmp/, luego solo habría que preocuparse por la carpeta /var/cache/apt/archives/, /var/log/ y /tmp/ haciendo más fácil y rápido todo el procedimiento.

La entrada de esta tarea son los sistemas bases actualizados y el responsable de ejecutarla es el desarrollador.

2.3.7 Creación de estructura de carpetas de la imagen de Nova

Para la realización de este paso se utilizarán los sistemas bases creados y la imagen de Nova Ligero. El responsable es el desarrollador.

Herramientas necesarias:

- syslinux: es un gestor de arranque para el sistema operativo Linux que opera desde un sistema de archivos FAT. Está pensado para simplificar la instalación por primera vez de Linux, y para la creación de rescates y otros discos de arranque de propósito especial.
- squashfs-tools: herramienta que permite comprimir y descomprimir una carpeta a formato squashfs el cual está pensado para su uso como sistema de archivo genérico de solo lectura y es el utilizado en Nova para comprimir el sistema base.
- genisoimage: es un programa para generar sistemas de archivos híbridos ISO9660 / Joliet / HFS. En el caso de Nova, ISO9660.
- xorriso: es un programa que copia objetos de archivos de sistemas de archivos compatibles con POSIX[15] en sistemas de archivos mejorados Rock Ridge ISO 9660[16] y realiza la manipulación de dichos sistemas de archivos en sesiones.

Se debe poseer una carpeta donde se realizará todo el proceso. En el siguiente ejemplo el nombre definido para dicha carpeta es “image”.

1. Copiar el núcleo del sistema base y el initrd que constituye el sistema de inicio de Linux.

[15] POSIX: es una familia de estándares especificados por la IEEE Computer Society para mantener la compatibilidad entre los sistemas operativos (ISO/IEC 9945, 2002).
[16] Rock Ridge ISO 9660: es una extensión del formato ISO 9660, comúnmente utilizado en CD-ROM y DVD, que agrega la semántica del sistema de archivos POSIX (ISO, 1988).

Para ellos primeramente hay que crear una carpeta llamada “casper” donde se almacenará dicha información *mkdir casper y* otras dos para almacenar las distribuciones de las dos arquitecturas *mkdir casper/i386 mkdir casper/amd64*, *cp chrooti386/boot/vmlinuz-**.**.**-generic image/casper/i386/vmlinuz cp chrootamd64/boot/vmlinuz-**.**.**-generic image/casper/amd64/vmlinuz* (núcleo)*, cp chrooti386/boot/initrd.img-**.**.**-generic image/casper/i386/initrd.lz cp chrootamd64/boot/initrd.img-**.**.**-generic image/casper/amd64/initrd.lz* (sistema de inicio)*.* Los asteriscos significan la versión del núcleo y del initrd que se haya instalado en el sistema base.

2. Crea el manifest que es un archivo que almacena el nombre de las aplicaciones instaladas en el sistema base *chroot chrooti386 dpkg-query -W --showformat='${Package} ${Version}\n' | sudo tee image/casper/filesystem.manifest.* Este archivo solo se creará una vez pues las aplicaciones para ambas arquitecturas son las mismas.
3. Realizar una copia del filesystem.manifest que será utilizada para almacenar el nombre de las aplicaciones que se desea tenga el sistema luego de instalado. La diferencia de ambos archivos es usada por el instalador para saber qué debe desinstalar luego de descomprimir el sistema en el computador *cp -v image/casper/filesystem.manifest image/casper/filesystem.manifest-desktop.*
4. Crear la variable REMOVE donde se especificará lo que se debe quitar de la copia realizada (.manifest-desktop). En el presente ejemplo se incluye en dicha variable las aplicaciones vinculadas con el instalador y el casper que posibilita el trabajo con el liveCD, pero estas aplicaciones dependen de la personalización que se desee realizar. REMOVE='ubiquity ubiquity-frontend-gtk ubiquity-frontend-kde casper lupin-casper live-initramfs user-setup discover1 xres-probe os-prober libdebian-installer4'.
5. Iterar sobre la variable REMOVE para ir eliminando las aplicaciones especificadas del .manifest-desktop for i in $REMOVE; do sed -i "/${i}/d" image/casper/filesystem.manifest-desktop; done.
6. Comprimir el sistema base con el formato .squashfs mksquashfs chrooti386 image/casper/i386/filesystem.squashfs mksquashfs chrootamd64 image/casper/amd64/filesystem.squashfs.
7. En este punto corresponde llenar la carpeta isolinux con la información requerida. Esta carpeta se encarga de almacenar configuraciones especiales que permiten el arranque del sistema. Teniendo en cuenta el cúmulo de información que se almacena en ella, se explicará en el punto 2.4.
8. Crear la carpeta “.disk” y dentro de ella un archivo llamado “info” con la siguiente información: *nova-escritorio 6.0 2017 – Release amd64 i386 (20180314)* donde nova-escritorio es el nombre de la distribución, 6.0 la versión, 2017 el codename (nombre de código), Release amd64 i386 la o las arquitecturas del sistema y entre paréntesis la fecha de liberación con

el formato año, mes, día, todo junto y sin caracteres entre ellos. También se crea un archivo llamado cd_type que almacena *full_cd/single* y un archivo llamado base_components que contiene las ramas del repositorio que almacenan la base, en el caso de Nova *principal*.

9. Crear repositorio con las aplicaciones que necesita el instalador para su correcto funcionamiento *aptly –filter="paquetes" mirror create livecd http://novarepo.uci.cu/nova 2017 principal,extendido, aptly mirror update livecd*. Donde los paquetes necesarios son: b43-fwcutter, bcmwl-núcleo-source, dkms, libc6-i386, grub, grub-efi, grub-efi-amd64, grub-efi-amd64-bin, grub-efi-amd64-signed, intel-microcode, lupin-support, iucode-tool, mokutil, moussemu, setserial, shim, shim-signed, oem-config, oem-config-gtk, oem-config-slideshow-ubuntu, user-setup, wvdial, libuniconf4.6, libwvstream4.6-base y libwvstream4.6-extras, además de todos los incluidos ya desde la estructura de carpeta de Nova Ligero. Este repositorio es necesario para la instalación en discos duros con tabla de partición GPT y no está descrito en el procedimiento de construcción de imágenes que utiliza Ubuntu, pero el propio estudio de la estructura de las imágenes de Ubuntu y del código del instalador utilizado permitió detectar la pertinencia del mismo.
10. Crear archivo con el tamaño de sistema antes de ser comprimido *printf $(sudo du -sx --block-size=1 chrooti386 | cut -f1) > image/casper/i386/filesystem.size printf $(sudo du -sx --block-size=1 chrootamd64 | cut -f1) > image/casper/amd64/filesystem.size*.
11. Crear archivo con la suma md5 de cada archivo almacenado en la imagen *cd image && find . -type f -print0 | xargs -0 md5sum | grep -v "\./md5sum.txt" > md5sum.txt*.

2.3.8 Prueba de aplicaciones innecesarias

Una aplicación innecesaria es aquella que por algún motivo fue instalada en el sistema y que por alguna razón ya no es necesaria. Tal es el caso de una dependencia, que fue instalada para poder satisfacer un requerimiento de una aplicación determinada y que esta última fue desinstalada con posterioridad. Este problema se puede detectar al ejecutar el comando *apt-get autoremove*. Cuando un sistema tiene solo lo que necesita este comando no hará nada, pero en caso contrario tratará de desinstalar todas las aplicaciones que tiene de más. De igual forma cuando se trata de instalar una aplicación nueva, si existen aplicaciones innecesarias, advertirá al respecto.

Se podrá pensar que con el comando anteriormente presentado se puede resolver el problema, pero para el desarrollador no es tan fácil. Una imagen de Nova posee dos archivos muy importantes: filesystem.manifest y filesystem.manifest-desktop. Estos archivos son utilizados por el instalador para, una vez instalado el sistema en un ordenador, desinstalar un conjunto de aplicaciones que ya no serán necesarias teniendo en cuenta las diferencias entre ellos, tal es el caso del propio instalador que no es necesario que esté presente en la estación de trabajo. Como se ha expresado anteriormente, cuando se instala una aplicación se instalan otras tantas que constituyen dependencias de la misma, entonces ¿cómo se pueden conocer las aplicaciones que deben ser

desinstaladas? Pues no existe un procedimiento definido, pero sí varios trucos que se pueden aplicar, aprovechando funcionalidades que brindan otras herramientas.

Si ya existe una imagen construida será necesario descomprimir el sistema base, montar los sistemas de ficheros necesarios y verificar la existencia de aplicaciones innecesarias.

unsquashfs –d chroot image/casper/filesystem.squashfs

mount –bind /dev chroot/dev

chroot chroot/

mount none –t proc proc

apt-get autoremove | grep ^

Esto mostrará el listado de aplicaciones innecesarias, o vacío en caso de que no haya ninguna. Estas aplicaciones deben ser eliminadas del archivo ffilesystem.manifest-desktop.

REMOVE='APLICACIONES_DETECTADAS'

for i in $REMOVE; do sed -i "/${i}/d" image/casper/filesystem.manifest-desktop; done

Para evitar este problema se propone el siguiente procedimiento:

Cuando se definen los requisitos funcionales y por tanto las aplicaciones que formarán parte del sistema, también se deberán definir las aplicaciones que deben formar parte del liveCD pero no del sistema instalado. En la etapa de instalación de aplicaciones en el sistema base se comenzará con todas las aplicaciones que estarán antes y después y una vez que se termine con ella se ejecutará apt-get clean, esto eliminará todos los .deb descargados por APT. A continuación, se instalarán el resto de las aplicaciones que sólo estarán en el liveCD, como puede ser ubiquity (instalador), casper, gparted, etc. Una vez que se concluye se deberá listar todos los .deb que se encuentran en /var/cache/apt/archives/ y estas aplicaciones son las que se deberán eliminar del archivo filesystem.manifest-desktop.

REMOVE = $(ls /var/cache/apt/archives/ | grep deb | cut –d “_” –f1)

for i in $REMOVE; do sed -i "/${i}/d" image/casper/filesystem.manifest-desktop; done

Con este procedimiento nunca quedarán aplicaciones innecesarias instaladas en el sistema por causa de los archivos manifest. La entrada de esta tarea es la estructura de carpetas de la imagen y el responsable es el desarrollador.

2.3.9 Creación de la imagen

Para construir la imagen se utiliza el siguiente comando el cual es compatible con el Creador de imágenes de Discos de Nova, herramienta oficial de la distribución para la creación de discos de arranque. *xorriso –as mkisofs –isohybrid-mbr isolinux/isohdpfx.bin -b isolinux/isolinux.bin –c*

isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table –eltorito-alt-boot –e boot/grub/efi.img –no-emul-boot –isohybrid-gpt-basdat -o "$PATH_TO_ISO_IMG" –r "." –sort-weight 0 / --sort-weight 1 /boot –joliet –joliet-long –volid $VOLNAME en sustitución del anteriormente utilizado y el propuesto por Ubuntu debido a su incompatibilidad con dicha herramienta en la versión 6.0 de la distribución (*mkisofs –r –V "$IMAGE_NAME" –cache-inodes –J –l –b isolinux/isolinux.bin –c isolinux/boot.cat –no-emul-boot –boot-load-size 4 –boot-info-table –o nova.iso .*) y que también permite introducirle los parámetros que posibilitan la instalación en modo UEFI definidos en el Capítulo 1 de la presente investigación.

Donde $PATH_TO_ISO_IMG es el lugar donde se desee almacenar la imagen y $VOLNAME el nombre que tendrá la misma.

Para realizar este paso es necesario contar con la estructura de carpetas de la imagen y se obtendrá la imagen del sistema. El responsable es el desarrollador.

2.3.10 Estrategia de soporte técnico para el proceso de migración

Durante el proceso de migración de Nova se detectan un conjunto de no conformidades que deben ser corregidas por el equipo de desarrollo y solucionadas lo antes posible. Para darle solución a las no conformidades es necesarios hacer modificaciones en los paquetes relacionados con la deficiencia y por tanto actualizar el repositorio de código fuente y binarios, lo que trae consigo el inicio de todo el procedimiento de construcción de imágenes de Nova Escritorio para de esta manera obtener un producto final con la no conformidad solucionada.

La entrada de este paso es la imagen de Nova y el responsable de realizar esta estrategia es el especialista de migración.

2.4 Isolinux

Primero es necesario copiar el binario de isolinux y de memtest:

cp /usr/lib/syslinux/isolinux.bin image/isolinux/

cp /boot/memtest86+.bin image/install/memtest

Para dar algunas instrucciones de arranque al usuario, se crea un archivo isolinux.txt en image/isolinux con la siguiente información de ejemplo:

splash.rle

****

Este es un CD de GNU/Linux Nova.

Para entrar al sistema por defecto presione en "live32bits". Para ejecutar memtest86+, presione "memtest"

****

Se puede mostrar un gráfico en el momento del arranque, pero es opcional. El presente ejemplo requiere un caracter especial junto con el nombre del archivo de la imagen “splash” (splash.rle). Para crear este caracter se ejecuta el siguiente comando:

printf "\x18" >emptyfile

Luego se edita el archivo vacío en cualquier editor de texto. Se agrega el nombre del archivo junto al primer caracter y se agrega el texto que se desea visualizar en el momento de inicio del sistema. Guarde el archivo como “isolinux.txt”.

Para crear el archivo splash.rle se crea una imagen de 480 píxeles de ancho. Se convierte a 15 colores, se indexa (tal vez usando GIMP) y se guarda como .bmp que convierte la imagen a un formato de mapa de bits. A continuación, se instala el paquete “netpbm” y se ejecuta

bmptoppm splash.bmp > splash.ppm

ppmtolss16 '#ffffff=7' < splash.ppm > splash.rle

Se crea el archivo isolinux.cfg en image/isolinux/ para proporcionar configuraciones del gestor de arranque. Se puede leer la información almacenada en /usr/share/doc/syslinux para conocer las opciones de configuración disponibles. A continuación, se muestra un ejemplo de lo que podría almacenar el archivo:

DEFAULT live32bits

LABEL live32bits

menu label ^Probar sin instalar Nova 6.0 32 bits

kernel /casper/i386/vmlinuz

append file=/cdrom/preseed/nova.seed boot=casper locale=es_ES initrd=/casper/i386/initrd.lz quiet splash --

LABEL live64bits

menu label ^Probar sin instalar Nova 6.0 64 bits

kernel /casper/amd64/vmlinuz

append file=/cdrom/preseed/nova.seed boot=casper locale=es_ES initrd=/casper/amd64/initrd.lz quiet splash –

LABEL live

menu label ^Probar sin instalar Nova Ligero 6.0

kernel /install/vmlinuz

append file=/cdrom/preseed/nova.seed boot=casper locale=es_ES initrd=/install/initrd.lz quiet splash --

LABEL check

menu label ^Check CD for defects

kernel /casper/amd64/vmlinuz

append boot=casper integrity-check initrd=/casper/amd64/initrd.lz quiet splash --

LABEL memtest

menu label ^Memory test

kernel /install/memtest

append -

LABEL hd

menu label ^Boot from first hard disk

localboot 0x80

append -

DISPLAY isolinux.txt

TIMEOUT 300

PROMPT 1

2.5 Diseño de la aplicación de software para el procedimiento

Para la materialización del procedimiento se decidió realizar una aplicación que cumpliera con cada uno de los pasos definidos.

2.5.1 Propuesta de solución

Se propone la creación de un software para la creación de las imágenes de Nova que interactúe directamente con las aplicaciones debootstrap, genisoimage, apt y squashfs-tool ordenándoles la creación de la base del sistema, la instalación de las aplicaciones necesarias, la realización de pruebas y la creación de la imagen. La pruebas permitirán ir detectando errores a los largo de todo el procedimiento que deben ser corregidos para garantizar la calidad del producto final. Este software será creado en forma de script y podrá ser ejecutado en la estación de trabajo del desarrollador.

2.5.2 Arquitectura del sistema

Para el desarrollo de la arquitectura, se tuvo en cuenta el estilo arquitectónico Tuberías y

filtros debido a que facilita en gran medida el desarrollo de la aplicación. En este estilo arquitectónico los componentes, llamados filtros, leen y procesan sus entradas para producir un flujo de salida. Existen diferentes tipos de filtros:

- Filtro fuente o source filter: producen un flujo de salida sin recibir ninguno de entrada.
- Filtro transformador o transform filter: recibe un flujo de entrada, lo procesa y genera un flujo de salida.
- Filtro consumidor o sink filter: recibe un flujo de entrada y no produce un flujo de salida.

Los conectores de procesos son las tuberías que llevan la salida de un filtro hacia la entrada del otro. Este estilo arquitectónico tiene muchas ventajas como la facilidad de ser descrito, entendido e implementado y su modularidad pues resultan irrelevantes los detalles de la implementación de cada uno de los filtros siempre que estén bien descritas sus entradas, salidas y objetivos (François, 2003).

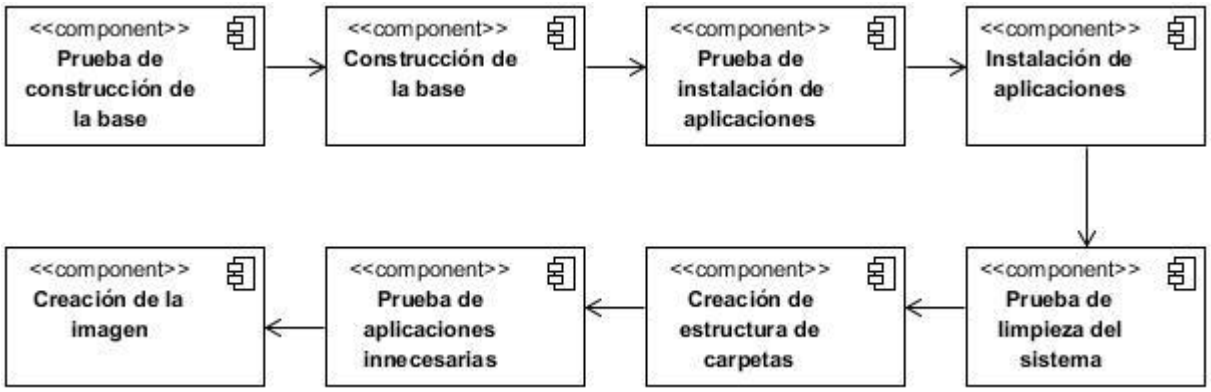

Figura 3: Arquitectura de la aplicación de construcción de imágenes de Nova.

En la presente arquitectura todo el flujo de datos se realiza sobre la base del sistema, siendo los filtros los encargados de procesarlo de acuerdo a lo establecido en el procedimiento expuesto anteriormente.

2.5.3 Requisitos funcionales

A continuación se detallan los requisitos funcionales del sistema.

Tabla 2: Requisitos funcionales de la aplicación de construcción de imágenes de Nova.

ID	Nombre	Descripción	Prioridad	Complejidad
RF1	Prueba de creación de la base.	Permite realizar una simulación de la creación de la imagen para determinar si no existen errores durante el proceso.	Media	Baja

RF2	Creación de la base del sistema.	Permite crear una base del sistema para las arquitecturas i386 y amd64.	Alta	Baja
RF3	Prueba de instalación de aplicaciones.	Permite instalar las aplicaciones definidas por el usuario en una base de prueba para determinar posible errores. Configura la aplicación APT para que devuelva la información necesaria.	Media	Alta
RF4	Instalación de aplicaciones	Permite instalar las aplicaciones definidas por el usuario.	Alta	Alta
RF5	Prueba de limpieza del sistema.	Permite realizar pruebas para identificar información innecesaria en el sistema base creado. Si detecta errores es capaz de corregirlos.	Media	Alta
RF6	Creación de estructura de carpetas de la imagen.	Permite crear la estructura de carpetas de la imagen tal cual se definió en la presente investigación, agrupando Nova Ligero, Nova Escritorio i386 y Nova Escritorio amd64.	Alta	Alta
RF7	Prueba de aplicaciones innecesarias.	Permite determinar si los archivos filesystem.manifest y filesystem.manifest-desktop contiene la información que deben contener para garantizar que el sistema no posea aplicaciones innecesarias	Alta	Alta

		una vez se instale en el ordenador.		
RF8	Creación de la imagen.	Permite obtener una imagen de Nova con las características definidas en la presente investigación.	Alta	Baja

2.6 Implementación

A partir de los resultados obtenidos se procede a materializar los mismos a código.

2.6.1 Lenguaje de programación

Bash es un intérprete de líneas de comando (o Shell) para el sistema operativo GNU y es compatible con sh e incorpora características de Korn Shell (ksh) y C Shell (csh) y fue desarrollado para cumplir con el estándar IEEE POSIX P1003.2 / ISO 9945.2 de Shell y Herramientas. Bash ofrece múltiples mejoras funcionales tanto para programación como de forma interactiva, entre las que se encuentran:

- Edición de la línea de comandos.
- Historial de comandos de tamaño ilimitado.
- Control de tareas.
- Arrays indexados de tamaño ilimitado.
- Aritmética de enteros en cualquier base desde dos hasta sesenta y cuatro.

Como casi todo el software de GNU, Bash es multiplataforma pudiéndose encontrar en casi todos los sistemas basados en UNIX y MS-DOS.

2.6.2 Estándar de codificación

Seguir un estándar de codificación consistente aumenta la calidad general de una aplicación de software y ayuda a la legibilidad de código escrito. Mientras más legible más fácil será para otra persona realizar acciones de mantenimiento o encauzar nuevos desarrollos y mejoras. Si a esto se le suma que resulta mucho más sencillo encontrar y corregir errores en un código normalizado, entonces seguir un estándar de codificación no es algo a tomar a la ligera.

Para la implementación del sistema de construcción de imágenes de Nova se sigue el estándar de codificación establecido por el lenguaje de programación Bash (Armstrong, 2015).

Conclusiones del capítulo

- La definición de una estructura que unifique las tres personalizaciones fundamentales de

Nova y la modificación de casper permitieron que las mismas puedan convivir en una única imagen.

- La descripción del procedimiento de construcción de la imagen de Nova posibilitó resolver un conjunto de problemas detectados con anterioridad como la incompatibilidad del sistema con la herramienta de construcción de imágenes de Nova y la instalación modo EFI además de servir como guía para los desarrolladores a la hora de construir una imagen.
- La creación de pruebas a realizar dentro del procedimiento de construcción de las imágenes en Nova posibilitó detectar un conjunto de deficiencias en el sistema y proponer medidas correctivas para solucionarlas.
- Se diseñó una aplicación en forma de script, utilizando el estilo arquitectónico Tuberías y filtros y el lenguaje de programación Bash lo que posibilitó ejecutar el procedimiento definido en el presente capítulo y obtener un producto que cumpliera con estas características.

CAPÍTULO 3. PRUEBAS

En el presente capítulo se realizan pruebas a las imágenes antes y luego de aplicar el procedimiento para determinar si se le da cumplimiento al objetivo de la investigación. Se aplican las pruebas de adecuación funcional, socio-adaptabilidad y se determina el índice de satisfacción grupal.

Variable independiente: procedimiento para la construcción de personalizaciones de Nova Escritorio.

Variables dependientes: adecuación funcional y socio-adaptabilidad.

3.1 Medidas de la calidad de adecuación funcional

Las medidas de la calidad de adecuación funcional se usan para evaluar el grado en que un producto o sistema proporciona funciones que satisfacen las necesidades declaradas e implícitas cuando se usan bajo condiciones específicas (Oficina Nacional de Normalización, 2017).

Nota 1: la adecuación funcional se refiere a si las funciones cumplen con las necesidades declaradas e implícitas.

Nota 2: la función podría ser un proceso elemental tal como se define en los requisitos funcionales del usuario en la ISO/IEC 14143.

Para evaluar la calidad de adecuación funcional se analizan varias subcategorías:

1. Calidad de completitud funcional.
2. Calidad de corrección funcional.
3. Calidad de pertinencia funcional.

Se utilizan los métodos de inspección ya que son métodos en donde un grupo de expertos examinan aspectos relacionados con el funcionamiento del software y la usabilidad de la interfaz. La Inspección de características fue el método seleccionado para evaluar las funcionalidades de Nova guiándose por el documento de especificación de requisitos y los casos de pruebas. La evaluación fue realizada por cinco evaluadores, 3 de ellos Máster y 2 Ingenieros, todos con más de 3 años de experiencia en el desarrollo de Nova o en el proceso de migración (Anexo 5).

3.1.1 Medidas de la calidad de completitud funcional

Las medidas de la calidad de completitud funcional se utilizan para evaluar el grado en que el conjunto de funciones cubre todas las tareas especificadas y los objetivos de los usuarios.

Tabla 3: Medidas de completitud funcional.

ID	Nombre	Descripción	Función de medición
FCp-1-G	Completitud	¿Qué proporción de las	X = 1 – A / B

	funcional	funciones especificadas se han implementado?	A = Número de funciones ausentes. B = Número de funciones especificadas.
Nota 1: las funciones se pueden especificar en una especificación de requisitos, una especificación de diseño, un manual de usuario o todos ellos. Nota 2: una función que falta se detecta cuando un producto de software o sistema no tiene la capacidad de realizar una función que es especificada.			

En la presente investigación se asumió como funciones cada uno de los requisitos definidos en el documento de especificación de requisitos del proyecto Nova 6.0. El mismo cuenta con 95 requisitos, no existe el RF79, RF90 y RF91, por lo que a pesar de que se puede apreciar la presencia del RF98, solo están especificados 95.

Siguiendo el procedimiento de Ubuntu para la construcción de imágenes

De los requisitos hay una que no se implementó:

1. RF98: Seleccionar variante de instalación.

Teniendo en cuenta esto se pueden definir las variables de la función de medición:

- A = 1
- B = 95

X = 1 – 1 / 95

X = 1 – 0, 0105

X = 0, 9895

Siguiendo el procedimiento definido en la investigación

- A = 0
- B = 95

X = 1 – 0/95 = 1

Por lo que se puede afirmar que la completitud funcional mejora con el uso del nuevo procedimiento.

3.1.2 Medidas de la calidad de corrección funcional

Las medidas de calidad de corrección funcional se utilizan para evaluar el grado en que un producto o sistema proporcionan los resultados correctos con el grado de precisión necesario.

Tabla 4: Medidas de corrección funcional.

ID	Nombre	Descripción	Función de medición

FCr-1-G	Corrección funcional	¿Qué proporción de las funciones proporciona los resultados correctos?	X = 1 – A / B A = Número de funciones incorrectas. B = Número de funciones consideradas.
Nota 1: una función incorrecta es aquella que no proporciona un resultado favorable y aceptable para lograr el objetivo específico deseado. Nota 2: las funciones consideradas para la evaluación pueden ser todas las funciones de un producto o un conjunto específico de funciones requeridas para un uso particular.			

En la presente prueba se consideran todas las funciones especificadas en el documento de especificación de requisitos del proyecto Nova 6.0.

Siguiendo el procedimiento de Ubuntu para la construcción de imágenes

De los requisitos hay tres que no proporcionan un resultado favorable:

1. RF20: Crear discos de arranque.
2. RF34: Pausar copia.
3. RF96: Instalar el sistema.

Teniendo en cuenta esto se pueden definir las variables de la función de medición:

- A = 3
- B = 95

X = 1 – 3 / 95

X = 1 – 0, 0316

X = 0, 9684

Siguiendo el procedimiento definido en la investigación

Hay un requisito que no proporciona un resultado favorable:

1. RF34: Pausar copia.

Teniendo en cuenta esto se pueden definir las variables de la función de medición:

- A = 1
- B = 95

X = 1 – 1/95 = 0, 9895

Por lo que se puede afirmar que la corrección funcional mejora con el uso del nuevo procedimiento.

3.1.3 Medidas de la calidad de pertinencia funcional

Las medidas de calidad de pertinencia funcional se utilizan para evaluar el grado en que las funciones facilitan el cumplimiento de tareas y objetivos específicos.

Tabla 5: Medidas de pertinencia funcional.

ID	Nombre	Descripción	Función de medición
FAp-1-G	Pertinencia funcional del objetivo de uso	¿Qué proporción de las funciones requeridas por el usuario proporciona un resultado apropiado para lograr un objetivo de uso específico?	X = 1 – A / B A = Número de funciones faltantes o incorrectas entre aquellas que son necesarias para lograr un objetivo de uso específico. B = Número de funciones necesarias para alcanzar un objetivo de uso específico.
Nota 1: esta función normalmente se considerará para los objetivos de uso más importantes o más frecuentes identificados. Por lo tanto, esta medida de la calidad se calcula primero para cada uno de los objetivos de uso definidos que se pueden perseguir en el sistema, y luego la siguiente medida de la calidad, es decir FAp-2-G "pertinencia funcional del sistema", se puede calcular colectivamente a través de todos los objetivos de uso para proporcionar una medida del sistema. Nota 2: los usuarios de esta Norma Internacional también podrían considerar la posibilidad de medir la proporción de objetivos de usuario que son alcanzables con el fin de obtener una mejor comprensión del impacto real en el uso previsto por el usuario.			
FAp-2-G	Pertinencia funcional del sistema	¿Qué proporción de las funciones requeridas por los usuarios para lograr sus objetivos proporciona resultados apropiados?	X = $\sum_{i=1}^{n} A_i/\mathrm{n}$ A_i = Puntuación de aptitud para el objetivo de uso i, es decir, el valor medio de FAp-1-G para el i-ésimo objetivo de uso específico. n = Número de objetivos de uso.

Los requisitos funcionales se pueden agrupar es los siguientes objetivos de uso teniendo en cuenta su funcionalidad:

1. Navegación por archivos del sistema (RF1, RF2, RF3, RF34).
2. Edición de textos y datos (RF6, RF7, RF8, RF9, RF10, RF33).
3. Aplicaciones especiales (RF15, RF16).
4. Instalación del sistema (RF20, RF89, RF96, RF98).
5. Internet (RF29, RF32, RF50, RF94, RF95, RF97).
6. Multimedia (RF5, RF35, RF36, RF37).

7. Dispositivos de hardware (RF12, RF17, RF18, RF19, RF22, RF45, RF46, RF47, RF48, RF51, RF52, RF62, RF63, RF64, RF92).
8. Gestión de software (RF21, RF23, RF24, RF25, RF26, RF27, RF55, RF65, RF66, RF67, RF68, RF69, RF70).
9. Copia de información (RF13, RF14, RF28, RF80, RF81, RF82, RF83, RF84, RF85, RF86).
10. Opciones avanzadas (RF11, RF30, RF87, RF88).
11. Configuraciones (RF38, RF39, RF40, RF41, RF42, RF44, RF49, RF53, RF54, RF56, RF61, RF93).
12. Ventanas (RF57, RF58, RF59, RF60).
13. Opciones generales (RF4, RF43, RF71, RF72, RF73, RF74, RF75, RF76, RF77, RF78).

Siguiendo el procedimiento de Ubuntu para la construcción de imágenes

Tabla 6: Pertinencia funcional del objetivo de uso con el procedimiento de Ubuntu.

<table>
<tr><td>Objetivo de uso</td><td>Requisitos faltantes o incorrectos</td><td colspan="2">Función de medición</td></tr>
<tr><td rowspan="2">Navegación por archivos del sistema</td><td rowspan="2">RF34: Pausar copia.</td><td>A = 1</td><td>B = 4</td></tr>
<tr><td colspan="2">X = 1 – 1 / 4 = 0,75</td></tr>
<tr><td rowspan="2">Edición de textos y datos</td><td rowspan="2"></td><td>A = 0</td><td>B = 6</td></tr>
<tr><td colspan="2">X = 1 – 0 / 6 = 1</td></tr>
<tr><td rowspan="2">Aplicaciones especiales</td><td rowspan="2"></td><td>A = 0</td><td>B = 2</td></tr>
<tr><td colspan="2">X = 1 – 0 / 2 = 1</td></tr>
<tr><td rowspan="2">Instalación del sistema</td><td rowspan="2">RF20: Crear discos de arranque.
RF96: Instalar el sistema.
RF98: Seleccionar variante de instalación.</td><td>A = 3</td><td>B = 4</td></tr>
<tr><td colspan="2">X = 1 – 3 / 4 = 1 – 0, 75 = 0, 25</td></tr>
<tr><td rowspan="2">Internet</td><td rowspan="2"></td><td>A = 0</td><td>B = 7</td></tr>
<tr><td colspan="2">X = 1 – 0 / 7 = 1</td></tr>
<tr><td>Multimedia</td><td></td><td>A = 0</td><td>B = 4</td></tr>
</table>

		X = 1 – 0 / 4 = 1	
Dispositivos de hardware		A = 0	B = 15
		X = 1 – 0 / 15 = 1	
Gestión de software		A = 0	B = 13
		X = 1 – 0 / 13 = 1	
Copia de información		A = 0	B = 10
		X = 1 – 0 / 10 = 1	
Opciones avanzadas		A = 0	B = 4
		X = 1 – 0 / 4 = 1	
Configuraciones		A = 0	B = 12
		X = 1 – 0 / 12 = 1	
Ventanas		A = 0	B = 4
		X = 1 – 0 / 4 = 1	
Opciones generales		A = 0	B = 10
		X = 1 – 0 / 10 = 1	

Luego la pertinencia funcional del sistema sería:

X = (0,75 + 1 + 1 + 0,25 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1) / 13 = 12 / 13 = 0,9231

Siguiendo el procedimiento definido en la investigación

Tabla 7: Pertinencia funcional de objetivo de uso.

Objetivo de uso	Requisitos faltantes o incorrectos	Función de medición	
Navegación por archivos del sistema		A = 1	B = 4
		X = 1 – 1 / 4 = 0,75	
Edición de textos y datos		A = 0	B = 6

		X = 1 – 0 / 6 = 1	
Aplicaciones especiales		A = 0	B = 2
		X = 1 – 0 / 2 = 1	
Instalación del sistema		A = 0	B = 4
		X = 1 – 0 / 4 = 1	
Internet		A = 0	B = 7
		X = 1 – 0 / 7 = 1	
Multimedia		A = 0	B = 4
		X = 1 – 0 / 4 = 1	
Dispositivos de hardware		A = 0	B = 15
		X = 1 – 0 / 15 = 1	
Gestión de software		A = 0	B = 13
		X = 1 – 0 / 13 = 1	
Copia de información		A = 0	B = 10
		X = 1 – 0 / 10 = 1	
Opciones avanzadas		A = 0	B = 4
		X = 1 – 0 / 4 = 1	
Configuraciones		A = 0	B = 12
		X = 1 – 0 / 12 = 1	
Ventanas		A = 0	B = 4

		X = 1 – 0 / 4 = 1	
Opciones generales		A = 0	B = 10
		X = 1 – 0 / 10 = 1	

Luego la pertinencia funcional del sistema sería:

X = (0,75 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1) / 13 = 12,75 / 13 = 0,9808

Por lo que se puede afirmar que la pertinencia funcional mejora con el uso del nuevo procedimiento.

3.2 Socio-adaptabilidad

Un gran reto de las TIC, acallado por los países ricos, es la eliminación de su carácter selectivo y elitista, que hoy extrapola las desigualdades y barreras del mundo real al espacio cibernético, generando lo que se ha dado en llamar “Brecha Digital”. Millones de personas en el mundo están muy distantes de convertirse en “internautas”, cuando aún no saben leer y escribir y su gran preocupación diaria es sobrevivir al hambre, la sed y las enfermedades. Solo se erradicará con la voluntad política de los gobiernos, la cooperación internacional y un mínimo de recursos de los que hoy el llamado mundo desarrollado despilfarra en publicidad, sobreconsumo o carrera armamentista (Valdéz, 2007).

En cuanto a infraestructura tecnológica, el bloqueo norteamericano no sólo impide la adquisición de equipamiento y programas informáticos desde compañías norteamericanas; por su carácter extraterritorial persigue las operaciones comerciales cubanas con empresas de otras nacionalidades, aún en las más distantes regiones (Valdéz, 2007).

Si bien es cierto que gracias a la revolución hoy se exhiben altos niveles de calidad en esferas asociadas al conocimiento, la ciencia y la técnica. El modelo consumista impuesto por el capitalismo ha hecho de las TIC el sector más convulso en lo que a obsolescencia planificada se refiere. Cuba, cumpliendo con el Programa Rector para la Informatización de la sociedad, mantiene una política de adquisición de tecnologías de última generación cada año, mas no va acorde con la ideología y modelo económico socialista el uso irracional de recursos naturales en beneficio de una economía de mercado.

Es entendible que en Cuba se utilice una extensa gama de equipamiento informático, que abarca desde ordenadores de última generación hasta ordenadores de más de 10 años de antigüedad. A esto se le suma las condiciones atípicas de conectividad a la red de redes, elemento fundamental en el desarrollo de aplicaciones y servicios (Fírvida, 2009).

Por lo que el reto para la industria nacional de software resulta mucho más elevado, al tener que desarrollar aplicaciones y servicios capaces de aprovechar el equipamiento informático declarado

obsoleto así como las tecnologías de punta que hay que asumir en aras del desarrollo de la sociedad cubana.

Teniendo en cuenta esto, se procede a seleccionar de forma aleatoria un conjunto de equipos de cómputos, fundamentalmente laptops de última generación, con Windows instalado por defecto y que hasta la creación del presente procedimiento resultaba muy difícil o imposible la instalación de Nova junto con Windows, imposibilitando la introducción del sistema operativo cubano de forma progresiva en las computadoras personales y dirigido fundamentalmente a su introducción en los hogares cubanos y no solo en las empresas y organismos estatales.

La presente prueba fue aplicada en el MINDUS. De un total de 97 laptops el sistema fue probado en 50 para un 51, 55% del total. Esta prueba fue realizada sobre los siguientes estratos:

- 6 Toshiba.
- 9 Dell.
- 7 HP.
- 9 Asus.
- 12 Lenovo.
- 7 Acer.

A continuación se detallan los modelos de las laptops probadas y los resultados de la prueba:

Tabla 8: Posibilidad de instalación antes y luego de la aplicación del procedimiento.

Computadora (Modelo)	Instalación antes del procedimiento	Instalación después del procedimiento
Toshiba Satelite CL45-C4370		X
TOSHIBA C45-C4205K		X
Toshiba Satellite Pro A50-D-124		X
Toshiba L775	X	X
Toshiba Tecra PS585U		X
Toshiba Satellite Click Mini	X	X
Dell Inspiron N4020	X	X
Dell Inspiron 15 5000 series		X
Dell Inspiron 15 3000 series		X

Dell Latitude 348		X
DELL INSPIRON 14 7460		X
Dell Inspiron 15 5567	X	X
Dell Inspiron 5577I541TBU		X
DELL INSPIRON 9T17G	X	X
Dell Latitud E3480	X	X
HP 15-BA020NS	X	X
HP 15-bs126ns		X
HP Pavilion Power 15		X
HP 14-am009la		X
HP 15-da0011la	X	X
HP 15-AC136LA		X
HP Envy M6-1215TX		X
Asus F540YA-XX046T		X
Asus ZenBook UX430U	X	X
Asus X541NA-GO613T	X	X
ASUS X541NA-GQ079T PQC N4200		X
ASUS UL80Vs		X
Asus K61IC-JX019V	X	X
Asus Zenbook UX305UA-FC005	X	X
ASUS Zenbook Flip UX360UA	X	X
Asus X441na-ga137		X
Lenovo IDEAPAD 110-15IBR		X

Lenovo Yoga 520		X
Lenovo Ideapad 310		X
Lenovo Thinkpad E470		X
Lenovo ThinkPad T580		X
Lenovo V330	X	X
Lenovo ThinkPad X1 Carbon		X
Lenovo Ideapad 510-15ISK		X
Lenovo Ideapad 320-15ISK		X
Lenovo Thinkpad T440		X
Lenovo LEN 110-15ISK		X
Lenovo 110-17ACL		X
Acer ES1-533-C140		X
Acer ES1-571-507Y	X	X
Acer ES1-523-2367		X
Acer ES1-731C5K9		X
Acer Aspire Timeline X 5830T	X	X
Acer Aspire 5336	X	X
Acer Aspire Switch 11 SW5-111-15QG		X

Las pruebas de instalación en estos dispositivos arrojaron la posibilidad de instalar 17 equipos de cómputo de los 50 probados, sin la necesidad de utilizar el procedimiento propuesto, lo cual representa el 34% del total. Muchas de estas instalaciones fueron posibles porque con anterioridad ya había sido cambiado el sistema operativo que traía por defecto el computador por motivos de actualización y la tabla de particiones del disco duro también había sido modificada. Los 50 equipos de cómputos probados permitieron la instalación de Nova luego de aplicado el procedimiento propuesto en el producto para un 100% del total.

Se puede concluir que luego de aplicado el procedimiento mejora la socio-adaptabilidad del producto, fundamentalmente amparada en la posibilidad de instalación en laptops de última generación.

3.3 Técnica de Iadov

El conocimiento del estado de satisfacción del usuario en cuanto a las imágenes de Nova que se despliegan en sus instituciones antes y luego de aplicar el procedimiento, es de gran utilidad para conocer si el procedimiento contribuye en el grado de satisfacción de los usuarios. La técnica de Iadov constituye una vía para el estudio del grado de satisfacción de los implicados en el procedimiento objeto de análisis.

La técnica V. A. Iadov en su versión original, fue creada por N. V. Kuzmina, para establecer el nivel de satisfacción por la profesión en carreras pedagógicas (Kuzmina, 1970). Luego algunos autores la han modificado en parte y aplicado, para valorar la satisfacción en múltiples campos y como parte de diagnósticos y validaciones en diferentes investigaciones (Fernández & López, 2014).

La técnica de Iadov se basa en el análisis de un cuestionario que tiene una estructura interna determinada, la cual sigue las relaciones que se establecen entre tres preguntas cerradas (cuya relación el sujeto desconoce) y el análisis posterior de dos preguntas abiertas. La relación entre las preguntas cerradas se establece a través del denominado "Cuadro Lógico de Iadov" (ver Tabla 8), indicando la posición de cada persona en la escala de satisfacción.

Las siguientes preguntas elaboradas por el autor fueron contempladas en el cuadro de Iadov, utilizado en esta investigación para medir la satisfacción de los usuarios del MINDUS sin la aplicación del procedimiento y del MINDUS, con la aplicación del procedimiento, teniendo en cuenta que en dicho ministerio se realizaron procesos de migración con la utilización de Nova 5.0 sin aplicar el procedimiento y Nova 5.0 luego de aplicado el procedimiento respectivamente.

Tabla 9: Cuadro lógico de Iadov.

5. ¿Le satisface la imagen de Nova que fue instalada en su estación de trabajo?	1. ¿Considera usted que sea correcto realizar un proceso de migración sin una imagen que se pueda adaptar a las estaciones de trabajo más modernas?								
	No			No se			Sí		
	3. ¿Si usted fuera a realizar un proceso de migración lo realizaría haciendo uso de la imagen de Nova utilizada en su institución?								
	Sí	No se	No	Sí	No se	No	Sí	No se	No
Me gusta mucho	1	2	6	2	2	6	6	6	6

No me gusta mucho	2	2	3	2	2	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	5
No sé qué decir	2	3	6	3	3	3	6	3	4

El número resultante de la interrelación de las tres preguntas indica la posición de cada sujeto en la escala de satisfacción. La escala de satisfacción es la siguiente:

- Clara satisfacción
- Más satisfecho que insatisfecho
- No definida
- Más insatisfecho que satisfecho
- Clara insatisfacción
- Contradictoria

Para obtener el índice de satisfacción grupal (ISG) se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y -1 como se muestra en la Figura 4.

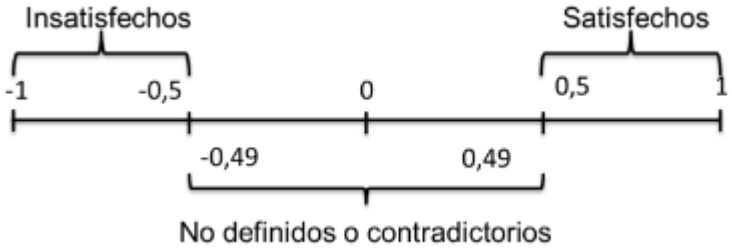

Figura 4: Escala numérica de Iadov.

La satisfacción grupal se calcula por la siguiente fórmula:

$$ISG = \frac{A(+1) + B(+0.5) + C(0) + D(-0.5) + E(-1)}{N}$$

En esta fórmula A, B, C, D, E, representan el número de sujetos con índice individual 1; 2; 3 ó 6; 4; 5 y donde N representa el número total de sujetos del grupo.

MINDUS antes del procedimiento

Para medir el grado de satisfacción se tomó una muestra de 24 usuarios, lo que constituye el 60% del total de usuarios implicados directamente en el proceso de migración en el MINDUS, a los cuales se les aplicó una encuesta (ver Anexo 1). Utilizando el cuadro lógico de Iadov definido anteriormente se recopilaron los resultados obtenidos de cada encuestado y se fue tomando la escala de satisfacción de cada resultado obtenido. El resultado del nivel de satisfacción según los encuestados se muestra en la Figura 5.

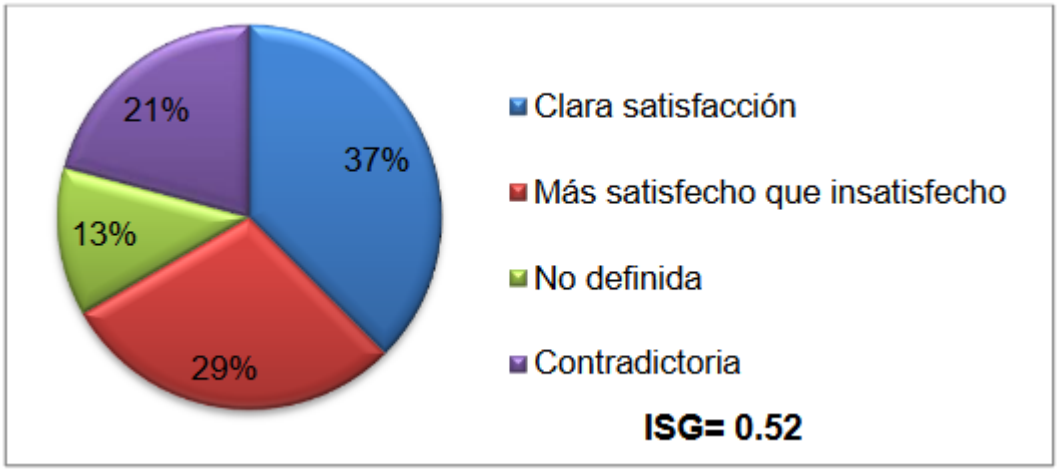

Figura 5: Resultado de la aplicación de la técnica de Iadov en el MINDUS antes de aplicar el procedimiento.

El índice grupal (ISG) arroja valores entre +1 y -1. Los valores que se encuentran comprendidos entre -1 y -0,5 indican insatisfacción; los comprendidos entre -0,49 y +0,49 evidencian contradicción y los que caen entre 0,5 y 1 indican que existe satisfacción. En este caso el valor del ISG según la fórmula planteada anteriormente fue de 0.52, lo que indica que los usuarios están satisfechos con imagen de Nova utilizada.

MINDUS luego de aplicado el procedimiento

Para medir el grado de satisfacción se tomó una muestra de 24 usuarios, lo que constituye el 60% del total de usuarios implicados directamente en el proceso de migración en el MINDUS, a los cuales se les aplicó una encuesta (ver Anexo 1).

Utilizando el cuadro lógico de Iadov definido anteriormente se recopilaron los resultados obtenidos de cada encuestado y se fue tomando la escala de satisfacción de cada resultado obtenido. El resultado del nivel de satisfacción según los encuestados se muestra en la Figura 6.

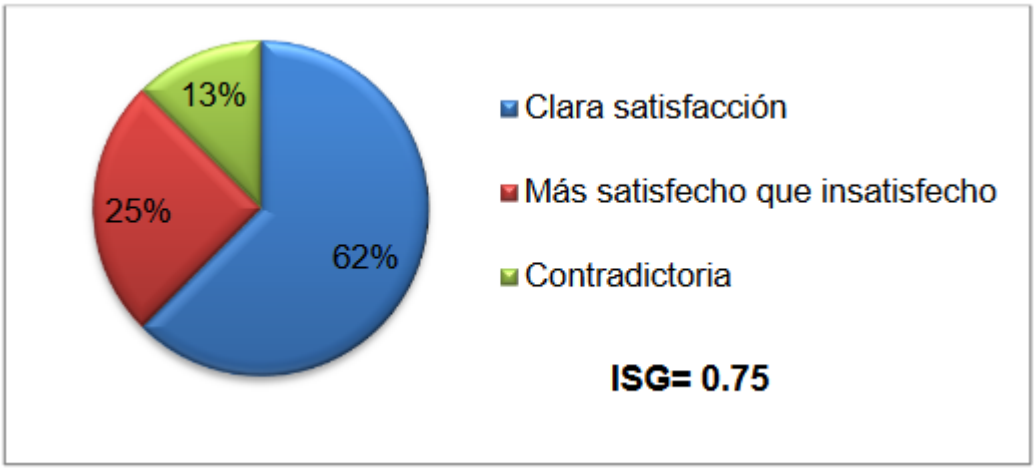

Figura 6: Resultado de la aplicación de la técnica de Iadov en el MINDUS luego de aplicar el procedimiento.

En este caso el valor del ISG según la fórmula planteada anteriormente fue de 0.75, lo que indica que los usuarios están satisfechos con imagen utilizada en el proceso, la cual fue realizada utilizando el procedimiento propuesto. Además, permite afirmar que se incrementó el grado de satisfacción de los usuarios con imagen brindada, pues el ISG fue mayor y que el porciento de usuarios que mostraron clara satisfacción fue el doble del anterior.

Luego de realizado el análisis comparativo se puede concluir que el procedimiento de construcción de imágenes de Nova, en su versión para escritorio contribuye con el aumento del grado de satisfacción de los usuarios con el producto instalado.

Conclusiones del capítulo

- La ejecución de pruebas de adecuación funcional posibilitó determinar que luego de la aplicación de la propuesta de solución la completitud, corrección y pertinencia funcionales mejoran por lo que se puede determinar que la adecuación funcional también lo hace.
- La ejecución de pruebas a varias laptops de última generación posibilitó determinar que luego de la aplicación del procedimiento propuesto fue posible la instalación de Nova en todas ellas sin la necesidad de crear una nueva tabla de particiones y en convivencia con Windows, demostrando un mejoramiento en los parámetros de socio-adaptabilidad.
- La ejecución de la técnica de Iadov para conocer el estado de satisfacción del usuario con el producto final antes y luego de la aplicación del procedimiento posibilitó determinar una mayor satisfacción con las imágenes luego de aplicado el mismo.

CONCLUSIONES GENERALES

- El estudio de las personalizaciones de GNU/Linux facilitó determinar las deficiencias de Nova, (como la imposibilidad de instalación en modo UEFI, de creación de memorias con la herramienta usb-creator, la inexistencia de pruebas dentro del proceso de desarrollo, la insatisfacción de los especialistas de migración por la dispersión de las imágenes) lo que posibilitó definir las posibles soluciones a esas problemáticas.
- La descripción del procedimiento para la construcción de imágenes de Nova Escritorio, posibilitó sistematizar el conocimiento tácito acumulado por los desarrolladores de la distribución cubana de GNU/Linux, incorporándole las modificaciones necesarias que permitieron la solución a las problemáticas detectadas.
- La aplicación del procedimiento propuesto facilitó la obtención de una imagen que agrupa las distribuciones de GNU/Linux Nova Escritorio en las arquitecturas i386 y amd64 así como Nova Ligero.
- La realización de pruebas de adecuación funcional y de socio-adaptabilidad sobre la imagen obtenida posibilitó determinar el mejoramiento de esas variables con la aplicación del procedimiento propuesto.

RECOMENDACIONES

- Mejorar el script que automatiza el procedimiento de construcción de la imagen.
- Incorporar el script en la herramienta de integración continua Jenkins, utilizada en el departamento de Sistemas Operativos.
- Definir el procedimiento para la construcción de las imágenes de Nova Ligero.

REFERENCIAS BIBLIOGRÁFICAS

Al Sadi, G. (2016). *Analyzing Master Boot Record for Forensic Investigations* (Vol. 10).

Armstrong, P. (2015). *Shell Style Guide*. Retrieved from https://google.github.io/styleguide/shell.%0Axml

Baig, R., & Aulí, F. (2003). *Sistema operativo GNU/Linux básico*.

Beekmans, G. (2010). *Linux From Scratch*. *Linux*. https://doi.org/10.1093/combul/41.3.32

Canonical. (2018). The leading operating system for PCs, IoT devices, servers and the cloud. Retrieved May 26, 2018, from https://www.ubuntu.com/

Chávez, H. R. Decreto Nº 3.390 - Decreto con Rango y Fuerza de Ley Orgánica de Ciencia, Tecnología e Innovación que obliga a la Administración Pública Nacional a emplear prioritariamente el Software Libre desarrollado con Estándares Abiertos (2004). Retrieved from http://www.wipo.int/edocs/lexdocs/laws/es/ve/ve052es.pdf

Correa, R. Decreto_1014_software_libre_Ecuador_c2d0b.pdf (2010). Retrieved from http://www.estebanmendieta.com/blog/wp-content/uploads/Decreto_1014_software_libre_Ecuador.pdf

Debian. (2018). Jenkins Debian. Retrieved from https://jenkins.debian.net/

El Senado y la Cámara de Representantes de la República Oriental del Uruguay. Ley 19.179 (2014). Uruguay. Retrieved from https://legislativo.parlamento.gub.uy/temporales/leytemp9117833.htm

Fallis, A. . (2013). *Understanding the LINUX KERNEL*. *Journal of Chemical Information and Modeling* (3rd ed., Vol. 53). O'Really. https://doi.org/10.1017/CBO9781107415324.004

Fernández, A., & López, A. (2014). Validación mediante criterio de usuarios del sistema de indicadores para prever, diseñar y medir el impacto en los proyectos de investigación del sector agropecuario. *Revista Ciencias Técnicas Agropecu Arias*, *23*, 77–82.

Fírvida, A. (2009). *Guano, entorno de escritorio cubano, libre y de código abierto. Trabajo de diploma para optar por el título de ingeniero en ciencias informáticas*. Universidad de las Ciencias Informáticas.

François, A. R. J. (2003). *Software Architecture for Computer Vision : Beyond Pipes and Filters*. *Computer*.

Gavasa, J. (2014, October). Software libre, cada vez más utilizado en Latino América.

González, R. (2007). Instalación de Linux en MacIntel. Retrieved from http://gro.usal.es/trabajos/RubenGonzalez/documentos/InstalacionLinuxMacIntel.pdf

Hobson, L. (2017). *System and method for using an extensible firmware interface (EFI) utility to build an EFI layer between an operating system and a legacy basic input/output system during a boot process*.

IEEE. (2018). IEEE 1003.1. Retrieved October 31, 2018, from http://pubs.opengroup.org/onlinepubs/9699919799/utilities/make.html#tag_20_76_13_04

ISO. (1988). ISO 9660: Information processing — Volume and file structure of CD-ROM for

information interchange. Retrieved March 14, 2018, from https://www.iso.org/obp/ui/#iso:std:iso:9660:ed-1:v1:en

ISO/IEC 9945. (2002). ISO/IEC and The Open Group announce international approval of the joint revision to POSIX and the Single UNIX Specification. Retrieved July 9, 2018, from https://www.iso.org/news/2002/11/Ref837.html

Jorba, J., & Suppi, R. (2004). *Administración avanzada de GNU/Linux*.

José, L., & González, S. (2009). Manual práctico de Linux con ejercicios. Retrieved from http://www.edu.xunta.gal/centros/iesfelixmuriel/system/files/manual_practico_de_linux_alumnos.pdf

Kuzmina, N. V. (1970). *Metódicas investigativas de la actividad pedagógica*. Leningrado.

Microsoft. (2014). Microsoft UEFI CA Signing policy updates – Windows Hardware Certification blog. Retrieved July 6, 2018, from https://blogs.msdn.microsoft.com/windows_hardware_certification/2013/12/03/microsoft-uefi-ca-signing-policy-updates/

Nikkel, B. J. (2009). *Forensic analysis of GPT disks and GUID partition tables*. *Digital Investigation* (Vol. 6). https://doi.org/10.1016/j.diin.2009.07.001

Nova. (2017). ¿Qué es Nova?

O&O Software. (2017). User's guide O&O DiskImage. Retrieved from https://www.oo-software.com/en/docs/usersguide/oodi12.pdf

Oficina Nacional de Normalización. (2017). *INGENIERÍA DE SOFTWARE Y SISTEMAS – REQUISITOS DE LA CALIDAD Y EVALUACIÓN DE SOFTWARE Y SISTEMAS (SQuaRE) – MEDICIÓN DE LA CALIDAD DEL PRODUCTO DE SOFTWARE Y DEL SISTEMA*. La Habana.

Organización Internacional de Estandarización. (2008). *Iso 9001*. https://doi.org/ISBN 978-92-67-10650-2

Partido Comunista de Cuba. Actualización de los lineamientos de la política económica y social del partido y la revolución para el periodo 2016-2021. (2016).

Pérez, Y., Goñi, A., García, A., García, J., Viera, A., Hernández, Y., … Pierra, A. (2015). *Guía cubana de migración a aplicaciones de código abierto*. La Habana.

Petersen, R., & Henrichsmeyer, D. (2008). *Linux: The Complete Reference*.

Pierre, A. (2015, February). *Nova, distribución cubana de GNU/Linux. Reestructuración estatégica de su proceso de desarrollo* (Maestría en Informática Aplicada). Universidad de las Ciencias Informáticas.

Pressman, R. S. (2013). *Ingeniería de Software un enfoque práctico*. *Journal of Chemical Information and Modeling* (Vol. 53). https://doi.org/10.1017/CBO9781107415324.004

Stallman, R. (2004). *Software libre para una sociedad libre*. *Http://Biblioweb.Sindominio.Net/Pensamiento/Softlibre/*. Retrieved from http://libros.metabiblioteca.org/handle/001/144%0Ahttp://bibliotecalibre.org/handle/001/144

Stallman, R. M. (2002). *Free software, free society : selected essays of Richard M. Stallman*. *Library Trends* (3rd ed.). Boston: Free Software Foundation. https://doi.org/10.1353/lib.2006.0005

Ubuntu. (2015). UEFI. Retrieved May 28, 2018, from https://help.ubuntu.com/community/UEFI

UEFI. (2014). *Unified Extensible Firmware Interface Specification*. Retrieved from http://www.uefi.org/sites/default/files/resources/2_4_Errata_B.pdf

Usb-creator. (2018). LinuxLive USB Creator. Retrieved July 8, 2018, from https://www.linuxliveusb.com/en/home

Valdéz, R. (2007). Cuba en la Cumbre Mundial sobre la Sociedad de la Información. Discurso pronunciado por el Comandante de la Revolución, Ramiro Valdés Menéndez, Ministro de la Informática y las Comunicaciones en el Acto Inaugural de la XII Convención y Expo Internacional,. Retrieved March 6, 2018, from http://anterior.cubaminrex.cu/Sociedad_Informacion/2007/DiscursoRamiro.htm

ANEXOS

Anexo 1: **Encuesta aplicada para determinar nivel de satisfacción**

Estimado(a) compañero(a):

Al contestar esta encuesta podrá dar a conocer su nivel de satisfacción con respecto a la imagen de Nova utilizada durante el proceso de migración a código abierto desarrollado recientemente en su institución. Se espera su sinceridad y se le agradece su colaboración en esta investigación:

1. ¿Considera usted que sea correcto realizar un proceso de migración sin una imagen que se pueda adaptar a las estaciones de trabajo más modernas?
   _____Sí _____No _____No sé
2. ¿Qué debería mejorarse en la imagen utilizada? Argumente.

   _____
3. ¿Si usted fuera a realizar un proceso de migración lo realizaría haciendo uso de la imagen de Nova utilizada en su institución?
   _____Sí _____No _____No sé
4. ¿En qué otras áreas de su institución cree pertinente realizar el proceso de migración?

   _____
5. ¿Le satisface la imagen de Nova que fue instalada en su estación de trabajo?
   _____Me gusta mucho.
   _____Me gusta más de lo que me disgusta.
   _____Me da lo mismo.
   _____Me disgusta más de lo que me gusta.
   _____No me gusta nada.

Gracias por su cooperación

Anexo 2: Entrevista a desarrolladores de GNU/Linux Nova

1. ¿Cómo se realiza el proceso de construcción de GNU/Linux Nova?

   _____

2. ¿En qué momento se realiza el primer ISO de la personalización?

   _____

3. ¿Resulta muy tediosa la construcción del ISO? ¿Cómo se hace?

   _____

4. ¿Solamente se realiza un ISO en el proceso de desarrollo?

   _____

5. ¿Dónde se almacenan los ISOs resultantes? ¿Hay algún encargado?

   _____

6. ¿En qué momento se pone público para la comunidad el ISO oficial de Nova?

   _____

7. ¿Puede tenerse acceso a este previamente aunque aún contenga errores?

   _____

8. ¿Qué ocurre cuando se necesita un ISO previamente hecho para agregarle o quitarle funcionalidades?

   _____

9. ¿Existe una guía oficial para la construcción de ISOs de GNU/Linux Nova?

   _____

10. ¿Quiénes poseen los conocimientos necesarios para construir un ISO?

    _____

11. ¿Los ISOs de Nova están listos para luego de instalarse poseer todas las aplicaciones necesarias que garanticen el cumplimiento de los requisitos funcionales?

    _____

12. ¿Cuáles son las principales deficiencias que poseen las imágenes de Nova?

    _____

13. ¿Está garantizada la seguridad de los ISOs que se realizan en el Departamento?

    _____

Gracias por su cooperación

Anexo 3: Entrevista a especialistas de migración del Departamento de Servicios Integrales de Migración, Asesoría y Soporte.

1. ¿Cómo se identifica la imagen a utilizar al migrar una estación de trabajo?

   _____

2. ¿Cuáles son las imágenes que más se utilizan en el proceso de migración?

   _____

3. ¿En qué dispositivos se realizan los booteables de estas imágenes?

   _____

4. ¿Qué capacidad de almacenamiento tienen estos dispositivos?

   _____

5. ¿Cuántos dispositivos utiliza cada migrador?

   _____

6. ¿Qué usted cree que sería provechoso realizar para que las imágenes sean más fáciles de manipular?

   _____

7. Exponga los principales elementos de las imágenes que entorpecen el proceso de migración.

   _____

Gracias por su cooperación

Anexo 4: Evaluación de los requisitos sin aplicar el procedimiento

Expertos:

- E1: Mtr. Yoandy Pérez Villazón (Jefe de Centro, Especialista de migración)
- E2: Mtr. María Leisy González Carrea (Jefa de proyecto HMAST, Especialista de Migración)
- E3: Mtr. Juan Manuel Fuentes Rodrígues (Jefe de Departamento, Desarrollador)
- E4: Ing. Nelio Veliz Pedraza (Jefe de Departamento, Especialista de Migración)
- E5: Ing. Yileni Hechevarría González (Jefa de proyecto Nova 6.0)

Los expertos evaluarán cada uno de los requisitos con puntuación 0 si el requisito no existe, 0,5 si el requisito existe pero tiene problemas y 1 si el requisito está correcto.

Tabla 10: Evaluación de los requisitos sin aplicar el procedimiento.

Requisitos / Especialistas	E1	E2	E3	E4	E5
RF1	1	1	1	1	1
RF2	1	1	1	1	1
RF3	1	1	1	1	1
RF4	1	1	1	1	1
RF5	1	1	1	1	1
RF6	1	1	1	1	1
RF7	1	1	1	1	1
RF8	1	1	1	1	1
RF9	1	1	1	1	1
RF10	1	1	1	1	1
RF11	1	1	1	1	1
RF12	1	1	1	1	1
RF13	1	1	1	1	1
RF14	1	1	1	1	1
RF15	1	1	1	1	1
RF16	1	1	1	1	1
RF17	1	1	1	1	1
RF18	1	1	1	1	1
RF19	1	1	1	1	1
RF20	0,5	0,5	0,5	0,5	0,5
RF21	1	1	1	1	1
RF22	1	1	1	1	1
RF23	1	1	1	1	1
RF24	1	1	1	1	1
RF25	1	1	1	1	1
RF26	1	1	1	1	1
RF27	1	1	1	1	1
RF28	1	1	1	1	1
RF29	1	1	1	1	1
RF30	1	1	1	1	1
RF31	1	1	1	1	1
RF32	1	1	1	1	1
RF33	1	1	1	1	1
RF34	0,5	0	0,5	0,5	0,5
RF35	1	1	1	1	1
RF36	1	1	1	1	1

RF37	1	1	1	1	1
RF38	1	1	1	1	1
RF39	1	1	1	1	1
RF40	1	1	1	1	1
RF41	1	1	1	1	1
RF42	1	1	1	1	1
RF43	1	1	1	1	1
RF44	1	1	1	1	1
RF45	1	1	1	1	1
RF46	1	1	1	1	1
RF47	1	1	1	1	1
RF48	1	1	1	1	1
RF49	1	1	1	1	1
RF50	1	1	1	1	1
RF51	1	1	1	1	1
RF52	1	1	1	1	1
RF53	1	1	1	1	1
RF54	1	1	1	1	1
RF55	1	1	1	1	1
RF56	1	1	1	1	1
RF57	1	1	1	1	1
RF58	1	1	1	1	1
RF59	1	1	1	1	1
RF60	1	1	1	1	1
RF61	1	1	1	1	1
RF62	1	1	1	1	1
RF63	1	1	1	1	1
RF64	1	1	1	1	1
RF65	1	1	1	1	1
RF66	1	1	1	1	1
RF67	1	1	1	1	1
RF68	1	1	1	1	1
RF69	1	1	1	1	1
RF70	1	1	1	1	1
RF71	1	1	1	1	1
RF72	1	1	1	1	1
RF73	1	1	1	1	1
RF74	1	1	1	1	1
RF75	1	1	1	1	1
RF76	1	1	1	1	1
RF77	1	1	1	1	1
RF78	1	1	1	1	1
RF80	1	1	1	1	1
RF81	1	1	1	1	1
RF82	1	1	1	1	1
RF83	1	1	1	1	1
RF84	1	1	1	1	1
RF85	1	1	1	1	1
RF86	1	1	1	1	1
RF87	1	1	1	1	1
RF88	1	1	1	1	1
RF89	1	1	1	1	1
RF92	1	1	1	1	1

RF93	1	1	1	1	1
RF94	1	1	1	1	1
RF95	1	1	1	1	1
RF96	0,5	1	0,5	1	0,5
RF97	1	1	1	1	1
RF98	0	0	0	0	0

Anexo 5: Evaluación de los requisitos luego de aplicar el procedimiento.

Expertos:

- E1: Mtr. Yoandy Pérez Villazón (Jefe de Centro, Especialista de migración)
- E2: Mtr. María Leisy González Carrea (Jefa de proyecto HMAST, Especialista de Migración)
- E3: Mtr. Juan Manuel Fuentes Rodrígues (Jefe de Departamento, Desarrollador)
- E4: Ing. Nelio Veliz Pedraza (Jefe de Departamento, Especialista de Migración)
- E5: Ing. Yileni Hechevarría González (Jefa de proyecto Nova 6.0)

Los expertos evaluarán cada uno de los requisitos con puntuación 0 si el requisito no existe, 0,5 si el requisito existe pero tiene problemas y 1 si el requisito está correcto.

Tabla 11: Evaluación de los requisitos luego de aplicar el procedimiento.

Requisitos / Especialistas	E1	E2	E3	E4	E5
RF1	1	1	1	1	1
RF2	1	1	1	1	1
RF3	1	1	1	1	1
RF4	1	1	1	1	1
RF5	1	1	1	1	1
RF6	1	1	1	1	1
RF7	1	1	1	1	1
RF8	1	1	1	1	1
RF9	1	1	1	1	1
RF10	1	1	1	1	1
RF11	1	1	1	1	1
RF12	1	1	1	1	1
RF13	1	1	1	1	1
RF14	1	1	1	1	1
RF15	1	1	1	1	1
RF16	1	1	1	1	1
RF17	1	1	1	1	1
RF18	1	1	1	1	1
RF19	1	1	1	1	1
RF20	1	1	1	1	1
RF21	1	1	1	1	1
RF22	1	1	1	1	1
RF23	1	1	1	1	1
RF24	1	1	1	1	1
RF25	1	1	1	1	1
RF26	1	1	1	1	1
RF27	1	1	1	1	1
RF28	1	1	1	1	1
RF29	1	1	1	1	1
RF30	1	1	1	1	1
RF31	1	1	1	1	1
RF32	1	1	1	1	1
RF33	1	1	1	1	1
RF34	0,5	0	0,5	0,5	0,5
RF35	1	1	1	1	1
RF36	1	1	1	1	1

RF37	1	1	1	1	1
RF38	1	1	1	1	1
RF39	1	1	1	1	1
RF40	1	1	1	1	1
RF41	1	1	1	1	1
RF42	1	1	1	1	1
RF43	1	1	1	1	1
RF44	1	1	1	1	1
RF45	1	1	1	1	1
RF46	1	1	1	1	1
RF47	1	1	1	1	1
RF48	1	1	1	1	1
RF49	1	1	1	1	1
RF50	1	1	1	1	1
RF51	1	1	1	1	1
RF52	1	1	1	1	1
RF53	1	1	1	1	1
RF54	1	1	1	1	1
RF55	1	1	1	1	1
RF56	1	1	1	1	1
RF57	1	1	1	1	1
RF58	1	1	1	1	1
RF59	1	1	1	1	1
RF60	1	1	1	1	1
RF61	1	1	1	1	1
RF62	1	1	1	1	1
RF63	1	1	1	1	1
RF64	1	1	1	1	1
RF65	1	1	1	1	1
RF66	1	1	1	1	1
RF67	1	1	1	1	1
RF68	1	1	1	1	1
RF69	1	1	1	1	1
RF70	1	1	1	1	1
RF71	1	1	1	1	1
RF72	1	1	1	1	1
RF73	1	1	1	1	1
RF74	1	1	1	1	1
RF75	1	1	1	1	1
RF76	1	1	1	1	1
RF77	1	1	1	1	1
RF78	1	1	1	1	1
RF80	1	1	1	1	1
RF81	1	1	1	1	1
RF82	1	1	1	1	1
RF83	1	1	1	1	1
RF84	1	1	1	1	1
RF85	1	1	1	1	1
RF86	1	1	1	1	1
RF87	1	1	1	1	1
RF88	1	1	1	1	1
RF89	1	1	1	1	1
RF92	1	1	1	1	1

RF93	1	1	1	1	1
RF94	1	1	1	1	1
RF95	1	1	1	1	1
RF96	0,5	1	1	1	1
RF97	1	1	1	1	1
RF98	1	1	1	1	1