Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

MODELO DE GESTIÓN DE ENTREGAS DE SOFTWARE PARA INCREMENTAR LA PRODUCTIVIDAD DEL EQUIPO DE DESARROLLO DEL PROYECTO DEBIAN GNU/LINUX

Josué Miguel Abarca Samayoa
Asesorado por el Ing. Joaquín Adolfo Guerrero Milián

Guatemala, septiembre de 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

MODELO DE GESTIÓN DE ENTREGAS DE SOFTWARE PARA INCREMENTAR LA PRODUCTIVIDAD DEL EQUIPO DE DESARROLLO DEL PROYECTO DEBIAN GNU/LINUX

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JOSUÉ MIGUEL ABARCA SAMAYOA
ASESORADO POR EL ING. JOAQUÍN ADOLFO GUERRERO MILIÁN

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, SEPTIEMBRE DE 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Kevin Vladimir Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
EXAMINADOR	Ing. Oscar Alejandro Paz Campos
EXAMINADOR	Ing. Everest Darwin Medinilla Rodríguez
SECRETARIO	Ing. Pablo Christian de León Rodríguez (a.i.)

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

MODELO DE GESTIÓN DE ENTREGAS DE SOFTWARE PARA INCREMENTAR LA PRODUCTIVIDAD DEL EQUIPO DE DESARROLLO DEL PROYECTO DEBIAN GNU/LINUX

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 12 de noviembre de 2020.

Josué Miguel Abarca Samayoa

Guatemala, 9 de abril de 2021.

Ing. Carlos Azurdia
Coordinador Proyectos de Graduación
Escuela de Ciencias y Sistemas
Facultad de Ingeniería

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante JOSUÉ MIGUEL ABARCA SAMAYOA, con carné 200212874 titulado: **"MODELO DE GESTIÓN DE ENTREGAS DE SOFTWARE PARA INCREMENTAR LA PRODUCTIVIDAD DEL EQUIPO DE DESARROLLO DEL PROYECTO DEBIAN GNU/LINUX"**, a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según protocolo propuesto, por lo que firmo la presente para que proceda con los trámites correspondientes.

Sin otro particular, me suscribo de usted.

Atentamente:

Joaquín Adolfo Guerrero Milián
Ingeniero en Ciencias y Sistemas
Colegiado No. 11,570
Asesor y revisor de trabajo de graduación

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala 23 de abril de 2021

Ingeniero
Carlos Gustavo Alonzo
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Alonzo:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **JOSUÉ MIGUEL ABARCA SAMAYOA** con carné **200212874 y** CUI **2337 53362 2101** titulado **"MODELO DE GESTIÓN DE ENTREGAS DE SOFTWARE PARA INCREMENTAR LA PRODUCTIVIDAD DEL EQUIPO DE DESARROLLO DEL PROYECTO DEBIAN GNU/LINUX"** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo aprobado.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"MODELO DE GESTIÓN DE ENTREGAS DE SOFTWARE PARA INCREMENTAR LA PRODUCTIVIDAD DEL EQUIPO DE DESARROLLO DEL PROYECTO DEBIAN GNU/LINUX",*** *realizado por el estudiante, JOSUÉ MIGUEL ABARCA SAMAYOA aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

Msc. Carlos Gustavo Alonzo
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 02 de septiembre de 2021

Decanato
Facultad de Ingeniería
24189101- 24189102

DTG. 395.2021

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado**: MODELO DE GESTIÓN DE ENTREGAS DE SOFTWARE PARA INCREMENTAR LA PRODUCTIVIDAD DEL EQUIPO DE DESARROLLO DEL PROYECTO DEBIAN GNU/LINUX,** presentado por el estudiante universitario: **Josué Miguel Abarca Samayoa,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANA
FACULTAD DE INGENIERÍA

Inga. Anabela Cordova Estrada
Decana

Guatemala, septiembre de 2021

AACE/cc

ACTO QUE DEDICO A:

Mis padres	Miguel Abarca y Argelia Samayoa, por sus enseñanzas, consejos y apoyo incondicional.
Mi esposa	Claudia Rosales, por creer en mí y porque juntos hemos alcanzado esta meta.
Mis abuelos	Por sus sabios consejos y por apoyarme en todo.
Mis hermanos	Por su ayuda y palabras de aliento.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser mi *alma máter* y brindarme la oportunidad de una formación superior.
Pueblo de Guatemala	Por darme la oportunidad de estudiar en esta gloriosa universidad.
Ing. Joaquín Guerrero	Por su confianza y el tiempo que ha dedicado al desarrollo de este proyecto de graduación.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

BTS — *Bug Tracking System* es el sistema de seguimiento de fallos del proyecto Debian que permite la creación de reportes de error y el control del trabajo de forma individual para cada paquete de software que forma parte del sistema Operativo Debian GNU/Linux.

COBIT — Marco de trabajo que provee un conjunto de mejores prácticas y procesos para la gobernanza de tecnologías de la información.

Código fuente — Conjunto de líneas de texto que contienen instrucciones que describen el funcionamiento de una tarea determinada para que sea ejecutada por una computadora.

Compilación — Proceso encargado de traducir de un archivo de código fuente con instrucciones entendibles por un programador hacia un archivo ejecutable con instrucciones entendibles por una computadora.

GNU — Es un proyecto colaborativo que publica los componentes básicos de un sistema operativo sin restricciones de uso, modificación o distribución.

GNU/Linux	Nombre formal para los sistemas operativos basados en el núcleo Linux y el conjunto de componentes desarrollados por el proyecto GNU.
Hardware	Son las partes físicas que conforman una computadora.
Internet	Conjunto de redes de computadoras interconectadas a nivel mundial para compartir información y recursos empleando un protocolo especial de comunicación.
ISO	Organización Internacional de Estandarización.
ITIL	Conjunto de mejores prácticas para la gestión de servicios de tecnologías de la información.
Linux	Nombre del núcleo creado en 1991 por Linus Torvalds.
Mejores prácticas	Es una compilación de las prácticas innovadoras que empresas reconocidas a nivel mundial han implementado, y que les han dado resultados positivos.
Núcleo	Es la parte del sistema operativo que se encarga de proveer acceso al hardware de la computadora.

Sistema operativo	Conjunto de programas que se encargan de realizar funciones básicas y permiten el desarrollo de otros programas.
Software	Conjunto de instrucciones que un computador procesa para realizar tareas específicas.

RESUMEN

Debian es una organización sin fines de lucro que desarrolla y da mantenimiento al sistema operativo Debian GNU/Linux, utilizado ampliamente en el ámbito empresarial, educativo y por organizaciones no gubernamentales. Fue fundado por Ian Murdock en 1993. El sistema operativo Debian GNU/Linux incluye paquetes de software que proveen una amplia gama de funcionalidades. El proyecto Debian GNU/Linux se rige por tres documentos que establecen su funcionamiento: la constitución de Debian, el contrato social de la comunidad y las directrices del software libre de Debian. La organización interna está compuesta por tres áreas principales de trabajo: coordinación, infraestructura y logística.

Al analizar la productividad del equipo de desarrollo de software desde el año 2009 hasta el 2019, utilizando los registros individuales de cada paquete de software que compone el sistema operativo Debian GNU/Linux, se observa que más del 40 % de paquetes de software reciben una actualización mayor en cada nueva versión, en ciclos de trabajo de 24 meses, con un crecimiento de más del 20 % en paquetes nuevos. Así mismo, se observa que la productividad del equipo de desarrollo ha tenido pérdidas de más del 30 % de trabajo en cada ciclo de desarrollo, debido a que la gestión de entregas de software actual solamente contempla la publicación de una nueva versión de Debian GNU/Linux luego de cumplirse un período mínimo, sin tomar en cuenta el nivel de productividad del equipo de desarrollo del software.

La gestión de entregas de software consiste en la planeación, el seguimiento y la administración de los procesos mediante los cuales se realiza la entrega de software al usuario final.

Con el paso del tiempo la industria de desarrollo de software ha definido múltiples estándares y marcos de trabajo que incluyen un conjunto de mejores prácticas para la gestión de entregas de software, como por ejemplo ITIL, COBIT, e ISO 2000, las cuales incluyen procesos para la gestión de aspectos como alcance, riesgo y frecuencia de las entregas, haciendo énfasis en la implementación de pruebas automáticas y manuales con la participación del cliente, con el objetivo de optimizar el trabajo realizado por el equipo de desarrollo, en el menor tiempo posible, maximizando la calidad del software entregado al usuario final.

El modelo para la gestión de entregas de software propuesto para el proyecto Debian GNU/Linux está compuesto por cuatro fases: verificación, validación, aceptación de usuario y producción. Estas fases incluyen un mecanismo para la evaluación continua del desempeño del equipo de desarrollo de software que permite optimizar el tiempo de entrega de nuevas funcionalidades al usuario final. El modelo también establece puntos de control para garantizar la calidad del software entregado, además de enfocarse en la detección temprana de errores y la gestión del riesgo de cada nueva entrega de software con el objetivo de mejorar la productividad del equipo de desarrollo de software.

OBJETIVOS

General

Diseñar un modelo de gestión de entregas de software para incrementar la productividad del equipo de desarrollo de software del proyecto Debian GNU/Linux.

Específicos

1. Describir desde una perspectiva de operación interna y recopilación documental, la organización y metodología de trabajo del proyecto Debian GNU/Linux.

2. Identificar los factores que afectan la productividad en la entrega de nuevas versiones de software del equipo de desarrollo del proyecto Debian GNU/Linux mediante análisis documental y obtención de métricas de los últimos 10 años.

3. Efectuar una recopilación documental de las mejores prácticas e infraestructura en la gestión de entregas de software para equipos de trabajo distribuidos geográficamente.

4. Definir un modelo aplicable al proyecto Debian GNU/Linux que permita incrementar la productividad del equipo de desarrollo de software mediante la automatización de procesos de integración, aprobación y entrega de software.

INTRODUCCIÓN

El enfoque de esta investigación está centrado en la metodología de trabajo que sigue el equipo de desarrollo de software del proyecto Debian GNU/Linux, la productividad individual y en conjunto, así como los factores que resultan en trabajo subutilizado, ocasionando un impacto negativo en su productividad. Los temas a tratar en esta investigación pretenden abordar dicha problemática y brindar una solución acorde a la organización considerando su flujo de trabajo actual.

El primer capítulo de la investigación, metodología de trabajo del proyecto Debian GNU/Linux, consistirá en una revisión de la estructura interna de la organización, explorando su jerarquía, misión y visión, las cuales están definidas en su constitución para luego dar a conocer las directrices y normas que rigen el trabajo de los voluntarios afiliados al proyecto.

En el segundo capítulo, factores que afectan la productividad en el proyecto Debian GNU/Linux, se analizará la productividad tanto en conjunto como de forma individual y también se analizarán los errores críticos detectados en cada paquete de software para culminar enumerando los factores que tienen un impacto negativo en la productividad del equipo de desarrollo de software.

En el tercer capítulo, mejores prácticas en la gestión de entregas de software, se analizarán los estándares ISO 20000, ANSI/PMI 99-001-2017 y COBIT, así como las mejores prácticas establecidas para la gestión de entregas continuas y DevOps, con la finalidad de identificar los elementos que permitan elaborar un nuevo modelo para la gestión de entregas de software que se adapte a las necesidades identificadas en el capítulo dos.

Finalmente, en el cuarto capítulo, modelo de gestión de entregas de software, se analizará el proceso que actualmente sigue el equipo de desarrollo del proyecto Debian/GNU Linux para luego identificar las oportunidades de mejora y proponer un modelo de gestión de entregas de software que, combinando los procesos y mejores prácticas expuestas en el capítulo tres, pueda sacar provecho de dichas oportunidades para mejorar la productividad en conjunto de todo el equipo de trabajo.

1. METODOLOGÍA DE TRABAJO DEL PROYECTO DEBIAN GNU/LINUX

Debian es una organización sin fines de lucro cuyo objetivo es el desarrollo y mantenimiento del sistema operativo Debian GNU/Linux[1] que es utilizado en el ámbito empresarial, educativo y por organizaciones no gubernamentales.[2] Debian fue creado en 1993 y fue bautizado por su fundador Ian Murdock, utilizando las primeras sílabas de su nombre y el de su esposa Debra Lynn, de ahí la palabra Debian[3].

El sistema operativo Debian GNU/Linux incluye un amplio catálogo de paquetes de software que van desde servidores HTTP, pasando por aplicaciones científicas, hasta software utilizado en el campo de la medicina[4] y ha mantenido un alto nivel de popularidad que lo llevó a ser elegido para ser utilizado por los astronautas a bordo de la Estación Espacial Internacional[5]. Es también la tercera distribución de Linux más utilizada en la plataforma de computación en la nube Amazon Web Services[6]. Debian GNU/Linux ha ocupado el puesto número seis en el ranking de popularidad de distribuciones de Linux de Distrowatch durante el 2020[7]. Por otro lado, Google Trends ubica a Debian GNU/Linux dentro de los

[1] SPI. *Constitución de Debian2016*. https://www.debian.org/devel/constitution.es.html.

[2] SPI. ¿Quién usa Debian? *2019*. https://www.debian.org/users/index.es.html.

[3] NIXON, Robin. *Ubuntu: up and running*.

[4] STROZZI, Francesco, JANSSEN, Roel, WURMUS, Ricardo, et. al. *Scalable Workflows and Reproducible Data Analysis for Genomics*. *Evolutionary Genomics*. p. 723-745.

[5] THE LINUX FOUNDATION. *Linux Foundation Training Prepares the International Space Station for Linux Migration*. https://training.linuxfoundation.org/solutions/corporate-solutions/success-stories/linux-foundation-training-prepares-the-international-space-station-for-linux-migration/.

[6] THE CLOUD MARKET. *EC2 Statistics*. https://thecloudmarket.com/stats#/by_platform_definition.

[7] ATEA ATAROA LIMITED. *DistroWatch Page Hit Ranking*. https://distrowatch.com/dwres.php?resource=popularity.

primeros cinco puestos de las distribuciones de Linux más populares de 2015 a 2020[8].

1.1. Contribuciones notables

> Ian Murdock fue el primer líder del proyecto, publicó la primera versión de Debian GNU/Linux el 1 de enero de 1994 y aunque desde su concepción se tenía como objetivo que el trabajo fuera distribuido entre un equipo de voluntarios, esta primera versión fue principalmente resultado del trabajo de Murdock. Durante 1994 el enfoque del proyecto fue hacer que el trabajo de los voluntarios se pudiera realizar de forma independiente y en paralelo, esto permitió que en noviembre de 1995 se publicara Debian GNU/Linux versión 0.93, fruto del trabajo de sesenta voluntarios, cada uno colaborando con el mantenimiento de uno o más paquetes de software9.

> Ian Murdock dejo de participar activamente en el proyecto Debian en marzo de 1996 nombrando nuevo líder del proyecto a Bruce Perens. Esta fue la primera ocasión que la organización experimento una transición de poder. Bruce, quién además de ser voluntario en el proyecto Debian GNU/Linux trabajaba como ingeniero en los estudios de animación Pixar[10], era conocido por su deseo de hacer que el software de código abierto ganara terreno en el sector empresarial.

> "Durante su gestión se empezaron a nombrar cada una de las versiones publicadas utilizando los personajes de la película Toy Story, coordino la creación del contrato social de Debian y la redacción de las directrices del software libre de Debian".[11]

Hartmuk Koptein, fue el principal desarrollador de la primera versión de Debian GNU/Linux con soporte para múltiples arquitecturas que fue publicada en 1998. Esta versión proveía soporte para procesadores Intel 386 y Morotola 68000[12], seguido de los procesadores Alpha y Sparc en marzo de 1999 y en el año 2000 las arquitecturas PowerPC y ARM. En este punto, el proyecto contaba

---

[8] GOOGLE LLC. *Google Trends*. https://trends.google.com/.
[9] GARBEE, Bdale, KOPTEIN, Hartmut, FERNÁNDEZ, Javier, et al. *A Brief History of Debian*. https://www.debian.org/doc/manuals/project-history/.
[10] IMDB INC. *Perfil de Bruce Perens*. http://www.imdb.com/name/nm0673302/.
[11] COLEMAN, E. Gabriella. *Three Ethical Moments in Debian*. p. 14.
[12] STARNES, Thomas. *Design Philosophy Behind Motorola's MC68000*. *Byte*.

con 450 integrantes y, con el objetivo de reunirse en persona y trabajar por el avance de la organización, se decidió instituir la reunión anual para desarrolladores del proyecto Debian conocida como Debconf. La primera reunión tuvo lugar del 5 al 9 de julio del 2000 en Burdeos, Francia.

Sam Hartman fue elegido líder del proyecto en el 2019 y bajo su liderazgo se publicó la versión diez del sistema operativo Debian GNU/Linux bajo el nombre clave, Buster el 6 de julio de 2019, contó con un equipo de trabajo constituido por más de mil voluntarios[13], soporta diez arquitecturas de hardware diferentes y tiene un catálogo con más de veintiocho mil paquetes de software[14].

1.2. Constitución de Debian

"La constitución de Debian fue creada en 1998 y establece la estructura organizacional para la toma formal de decisiones. La constitución garantiza los siguientes derechos a todo miembro del proyecto"[15]:

- Tomar cualquier decisión técnica o no técnica con relación a su propio trabajo
- Proponer o patrocinar proyectos de resolución general
- Proponerse a sí mismo como candidato para el puesto del líder del proyecto
- Votar en resoluciones generales y en las elecciones del líder del proyecto

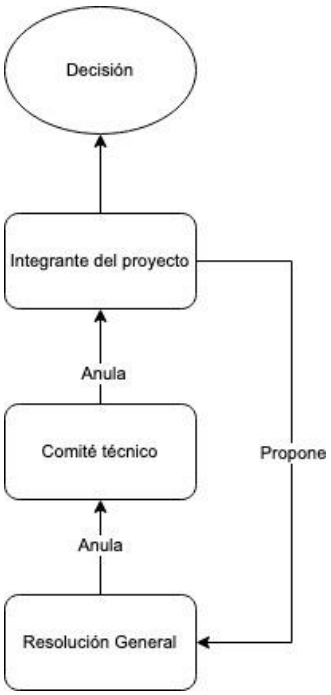
La constitución de Debian también pone de manifiesto el carácter voluntario del proyecto dejando claro que, nada en la constitución impone la obligación a

[13] SPI. *Status Debian Developer*. https://nm.debian.org/public/people/dd_all/.
[14] *IBID*. *Debian 10 Buster publicado*. https://www.debian.org/News/2019/20190706.es.html.
[15] SPI. *Constitución de Debian*. https://www.debian.org/devel/constitution.es.html.

nadie de realizar un trabajo para el proyecto, es decir que, una persona que no quiere realizar una tarea que le ha sido asignada, no está obligada a hacerla. Sin embargo, nadie debe trabajar activamente en contra de las normas y decisiones que han sido tomadas. De esta cuenta, una persona puede dimitir del puesto que posee o abandonar la organización con solo declararlo públicamente.

La constitución de Debian establece tres formas para la toma de decisiones: el punto de vista del desarrollador que está a cargo de la tarea; el comité técnico y la resolución general. La jerarquía entre los mecanismos se ilustra en la siguiente figura.

Figura 1. **Mecanismos para la toma de decisiones**



Fuente: elaboración propia, empleando Drawio.

> Las decisiones tomadas por el desarrollador a cargo de la tarea suelen ser la norma lo que permite que el trabajo fluya en el día a día, sin embargo, cuando estas decisiones afectan el trabajo de otros integrantes del proyecto suelen haber discusiones que se resuelven, en su mayoría, por consensos basados en argumentos técnicos, cuando no se logra alcanzar un consenso el desarrollador a cargo del paquete toma la decisión y si alguien no está de acuerdo puede solicitar la intervención del comité técnico[16].

El comité técnico de Debian, definido en la sección 6 de la constitución, es un grupo de personas que son electas por el líder del proyecto en conjunto con otros miembros del comité técnico y cuyo trabajo consiste en tomar la decisión final cuando surgen disputas técnicas entre otros voluntarios del proyecto.

> Antes de solicitar la intervención del comité técnico se espera que las partes involucradas hayan tenido una discusión constructiva en la que se abordan los puntos de vista de todos los interesados; los integrantes del comité técnico discuten los argumentos presentados y tratarán de tomar una decisión tan pronto como les sea posible, en algunos casos puede tomar varias semanas. Es muy común que, durante la discusión pública entre los miembros del comité, alguna de las partes en conflicto cambie de opinión y acepte el punto de vista del otro, esto es considerado positivo dentro del proyecto porque en dichos casos el comité técnico ya no toma la decisión final y el conflicto se da por solucionado[17].

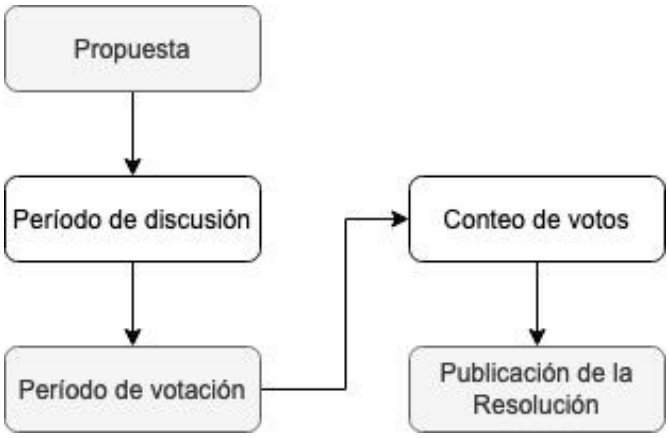
> El mecanismo llamado resolución general, es la instancia con mayor jerarquía para la toma de decisiones, cualquier integrante del proyecto Debian puede proponer una resolución general la cual debe ser apoyada por al menos otros cinco integrantes del proyecto, luego de ser propuesta se inicia el período de discusión en el cual otros integrantes pueden argumentar a favor o en contra y también proponer opciones alternativas. Finalizado este período se procede a la votación y la opción ganadora se convierte en la decisión que el proyecto adopta en conjunto. El secretario del proyecto, el cual es nombrado directamente por el líder del proyecto, está a cargo de organizar el proceso de votación para las resoluciones generales. En la siguiente figura se detallan las etapas que sigue una resolución general[18].

[16] COLEMAN, E. Gabriella, *Three Ethical Moments in Debian.* p. 10.

[17] SPI. *Comité técnico de Debian*. 2020, https://www.debian.org/devel/tech-ctte.es.html.

[18] SPI. *Procedimiento habitual de resoluciones*. https://www.debian.org/vote/howto_follow.es.html.

Figura 2. **Resolución general**



Fuente: elaboración propia, empleando Drawio.

La gestión de bienes también se encuentra definida en la constitución de Debian donde se establece que se debe recurrir a una organización de confianza para que sea la propietaria legal de todos los bienes del proyecto Debian. Actualmente la organización llamada Software In The Public Interest posee la propiedad legal de los dominios de: Internet, marcas registradas, cuentas de donaciones monetarias y equipos de hardware del proyecto Debian. Esto permite una gestión clara de los bienes porque ambas organizaciones operan de forma independiente y cada vez que se necesita hacer uso de los recursos que corresponden a Debian, se realiza a través de una solicitud pública de manera que quede debidamente registrado. La organización de confianza no posee ninguna autoridad sobre las decisiones y el funcionamiento del proyecto Debian[19].

El líder del proyecto Debian, definido en la sección 5 de la constitución, es un puesto al que todos los integrantes del proyecto pueden optar y es asignado anualmente mediante una votación que es coordinada por el secretario del proyecto. El líder del proyecto tiene a su cargo varias funciones: actúa como representante en conferencias y otros eventos externos al proyecto; dentro de la comunidad Debian, se encarga de coordinar y facilitar discusiones haciendo su mejor esfuerzo para resolver conflictos y asegurarse de que los mecanismos para la toma de decisiones estén funcionando de forma apropiada y acorde a la constitución. También está a cargo del nombramiento de los puestos del secretario del proyecto y los miembros del comité técnico y de comunicarse con la organización de confianza para notificar las decisiones respecto al uso de los bienes del proyecto Debian[20].

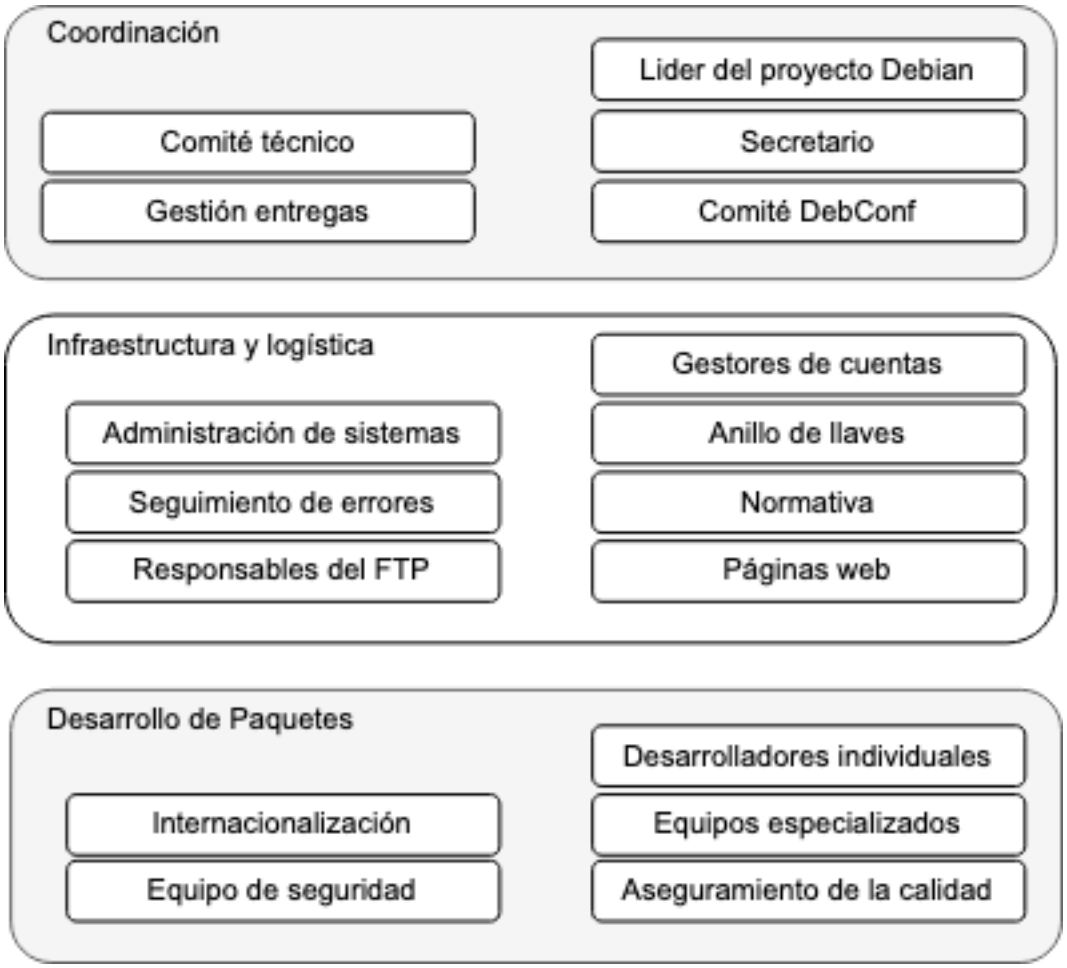
[19] SPI. *Debian - Donaciones*. *2020.* https://www.debian.org/donations.es.html.
[20] SPI. *Debian. 2020.* https://www.spi-inc.org/projects/debian/.

1.3. Organización Interna

En sus inicios el grupo de menos de veinticuatro voluntarios que formaban parte del proyecto Debian trabajaban de una forma no estructurada realizando tareas a su conveniencia, pero, con el paso del tiempo, el proyecto tuvo que adaptarse al aumento de voluntarios y de trabajo. Entre 1997 y 1999 surgieron las bases que definen la organización interna en la actualidad[21]. Se pueden identificar tres áreas de trabajo: coordinación, infraestructura y logística, y desarrollo de paquetes. Las responsabilidades están asignadas a voluntarios de forma grupal o individual. En la siguiente ilustración se detalla la conformación de cada área de trabajo[22].

[21] COLEMAN, E. Gabriela, *Three Ethical Moments in Debian.* p.11.
[22] SPI. *Organigrama de Debian*. *2020*. https://www.debian.org/intro/organization.es.html.

Figura 3. **Áreas de trabajo**



Fuente: elaboración propia, empleando Drawio.

1.3.1. Coordinación

El área de trabajo de coordinación está definida explícitamente en la constitución del proyecto. Está a cargo del líder del proyecto Debian y es un puesto por elección popular que se lleva a cabo cada año con todos los integrantes que están registrados oficialmente. El líder del proyecto nombra a los

delegados que integran el comité técnico, el equipo de gestión de entregas, el comité Debconf y el secretario del proyecto.

El comité Debconf está a cargo de la organización de la conferencia anual que sirve de punto de encuentro físico para los equipos de desarrollo. Durante la conferencia se suelen tener mesas de discusión para la toma de decisiones y se realizan sesiones de trabajo en equipo. Los puestos y equipos de trabajo que conforman el área de coordinación serán tratados en la sección titulada constitución de Debian.

1.3.2. Infraestructura y logística

La infraestructura y logística está a cargo de múltiples equipos que prestan servicios al resto de desarrolladores en el proyecto, el equipo de gestores de cuentas tiene a su cargo el determinar o decidir quién es un integrante oficial del proyecto y tiene la potestad de expulsar a alguien en casos específicos por mal comportamiento en la comunidad; este equipo trabaja junto al equipo de administradores del sistema para notificarles quién debe tener acceso a los distintos servidores que forman parte de la infraestructura y con el equipo que administra el anillo de llaves para darle acceso a la publicación de paquetes de software a los integrantes oficiales.

El equipo conocido como responsables del FTP, administra los servidores que almacenan todos los paquetes de software que conforman el sistema operativo Debian GNU/Linux y trabajan en conjunto con el equipo que administra el anillo de llaves para firmar cada paquete binario antes de su distribución a los usuarios finales.

El equipo a cargo de la normativa Debian trabaja con todos los desarrolladores para establecer las mejores prácticas y estándares de calidad que todos los paquetes de software deben cumplir, una de las tareas a su cargo es apoyar en el desarrollo de herramientas que verifiquen de forma automática el cumplimiento de la normativa Debian.

1.3.3. Desarrollo de paquetes

> Cada voluntario se postula así mismo para: formar parte de un equipo especializado, ocupar una posición o cumplir con una tarea. Debido a que la unidad básica que constituye el sistema operativo Debian GNU/Linux es la de paquetes de software, la mayoría de las personas trabajan en el área de desarrollo de paquetes. Esto fue una decisión intencional de los fundadores del proyecto que diseñaron el sistema de paquetes y la normativa que lo rige con el objetivo de potenciar la colaboración y facilitar la integración del trabajo distribuido entre los voluntarios alrededor del mundo. Es decir que el sistema está dividido en piezas individuales que permiten a distintas personas trabajar de forma independiente en ellas y al mismo tiempo permiten tener una forma clara de combinarlas para obtener un sistema integrado[23].

> La coordinación entre desarrolladores se lleva a cabo utilizando el sistema de seguimiento de errores de Debian o Bug Tracking System (BTS por sus siglas en inglés). El BTS es un sistema que permite la creación de reportes de error y el control del trabajo de forma individual para cada paquete que forma parte del sistema Debian GNU/Linux. Cada tarea que se realiza en el desarrollo de paquetes inicia con un registro en el BTS y como se ilustra en la figura 4 también se registran los avances, reportes de errores y solicitudes de cambios[24].

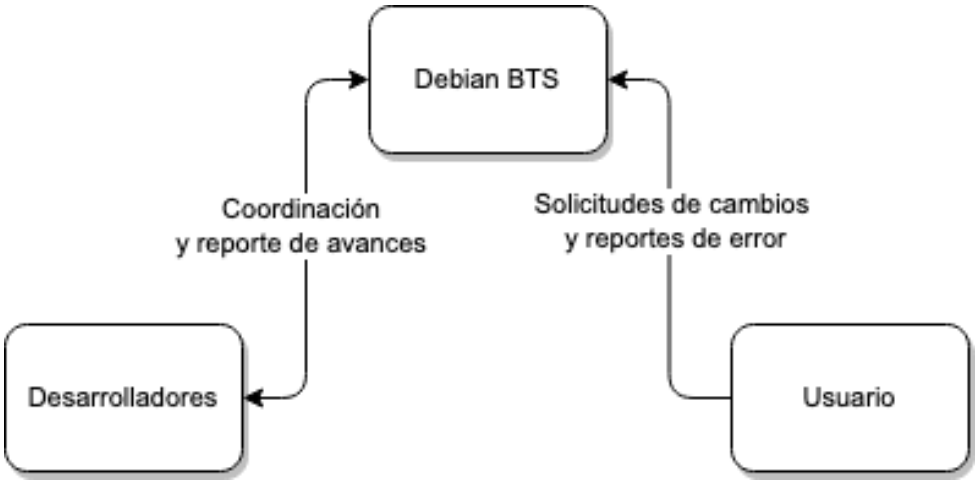
El BTS es desarrollado por el equipo de infraestructura llamado: equipo de seguimiento de errores de Debian. El BTS es uno de los paquetes que es desarrollado al 100 % por Debian y no se basa en código fuente externo; está pensado para las necesidades y flujos de trabajo del equipo de desarrollo de software del proyecto Debian GNU/Linux.

---

[23] COLEMAN, Gabriella. *Debian History Roundtable Discussion*. https://gabriellacoleman.org/debian-history-roundtable-discussion/.

[24] JACKSON, Ian y BENHAM, Darren. *Debian BTS*. https://www.debian.org/Bugs/server-control.es.html.

Figura 4. **Coordinación utilizando el BTS**



Fuente: elaboración propia, empleando Drawio.

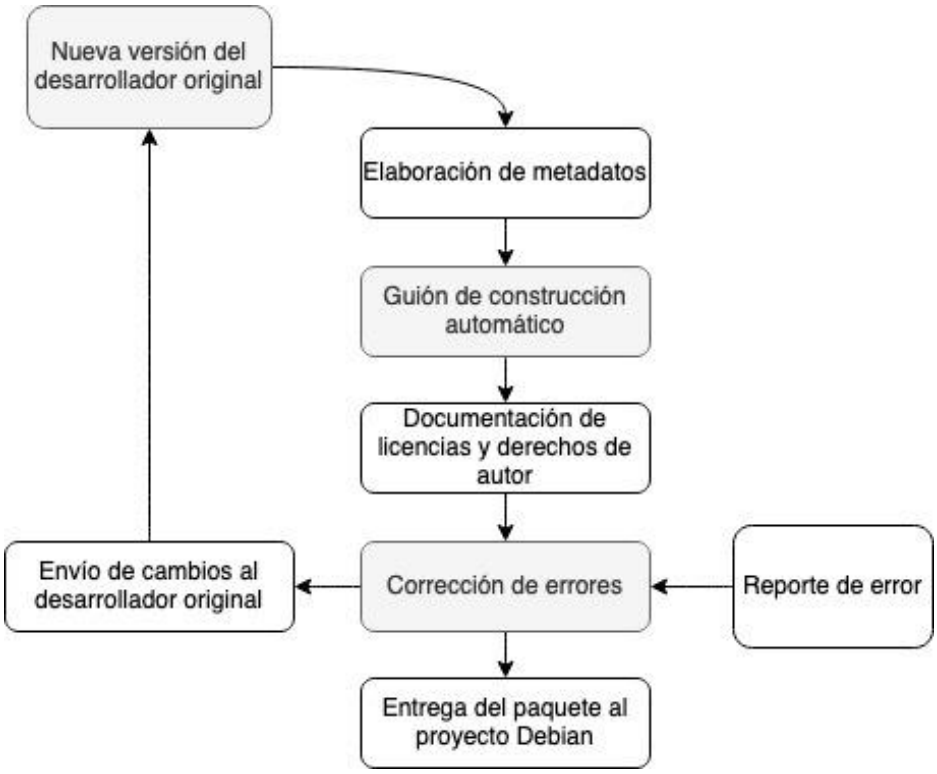
Los voluntarios pueden trabajar como desarrolladores individuales o formar equipos especializados con un interés común y cada paquete de software tiene asignado por lo menos un desarrollador responsable. La mayor parte del software incluido en Debian es creado por otros desarrolladores de software de forma independiente. El ciclo de trabajo de un desarrollador del equipo Debian, representado en la siguiente figura, conlleva: la definición de los metadatos para la integración con el resto de los paquetes de software que existen en el sistema; el desarrollo de los guiones de construcción automática; la documentación de licencias y derechos de autor; la corrección de errores y mejoras en el código fuente[25].

El equipo de aseguramiento de la calidad se encarga de proporcionar actualizaciones y resolver errores en todos los paquetes que quedan huérfanos cuando el desarrollador asignado ya no tiene la capacidad de continuar con sus responsabilidades.

Existen dos criterios para considerar a un paquete huérfano: 1. Cuando el desarrollador que lo tenía a su cargo explícitamente declara que no puede continuar dedicándole tiempo al cuidado del mismo; 2. Cuando un paquete de software tiene muchos reportes de errores sin resolver y no recibe actualizaciones durante un período de varios meses.

[25] LEVSEN, Holger, YAMANE, Hideki, NUSSBAUM, Lucas, et al. *Debian Developer's Reference*. https://www.debian.org/doc/manuals/developers-reference/index.en.html.

Figura 5. **Ciclo de trabajo**



Fuente: elaboración propia, empleando Drawio.

El equipo de seguridad está a cargo del manejo de errores que afectan la seguridad de los usuarios, cuando se recibe un reporte este equipo trabaja en conjunto con el desarrollador responsable del paquete de software afectado de manera que la solución sea publicada en el menor tiempo posible, también se encarga de hacer pública toda la información relacionada con el incidente para que los usuarios estén informados de la actualización de seguridad disponible.

El equipo de internacionalización se encarga de brindar soporte para múltiples idiomas y formatos locales para múltiples países. Actualmente Debian está disponible en setenta y cinco lenguajes para los cuales el equipo lleva a cabo las traducciones. El nivel de localización es variado e incluye aspectos como: formato de fecha, moneda, formato numérico, estándares de tamaños de página, entre otros y depende normalmente de la capacidad del equipo a cargo del idioma el poder cumplir con todos los aspectos[26].

[26] SPI. *Status of the l10n in Debian*. https://www.debian.org/international/l10n/po/rank.

1.4. Contrato social de la comunidad

> Este documento consiste en una serie de compromisos públicos a los que se adhieren todos los integrantes del proyecto Debian GNU/Linux. La idea de este documento fue propuesta por Ean Schuessler luego de una conversación con Bob Young, uno de los fundadores de la empresa Red Hat, que se dedica al desarrollo de un sistema operativo basado en Linux. Bob hizo ver que consideraba perjudicial para su empresa el comprometerse a producir exclusivamente software de código abierto y esto motivo a Ean a buscar que el proyecto Debian se distinguiera de otros proyectos similares comprometiéndose por escrito a crear exclusivamente este tipo de software[27].

> El contrato social con la comunidad cumple la función de definir claramente los compromisos del proyecto Debian a nuevos integrantes, usuarios del sistema operativo y personas ajenas al proyecto. Este documento fue redactado en 1997 bajo la coordinación de Bruce Perens y con la contribución de todos los integrantes de la organización. El 25 de abril de 2004 se aprobó por mayoría de votos la resolución general 2004-03[28] agregando varios cambios a la redacción del contrato social para aclarar que otros tipos de archivos como la documentación también deben cumplir con los requerimientos definidos en las directrices del software libre de Debian previo a formar parte del sistema operativo Debian GNU/Linux[29].

1.4.1. Licencia de Debian GNU/Linux

La sección titulada: Debian permanecerá 100 % libre, establece los criterios que la licencia de un paquete de software debe llenar para poder formar parte del sistema operativo Debian GNU/Linux. Establece que un software que ha sido desarrollado por un tercero y se desea integrar de manera oficial como parte de Debian GNU/Linux, la licencia del software debe cumplir con las directrices del software libre de Debian las cuales se tratarán a profundidad más adelante.

---

[27] COLEMAN, E. Gabriella. *Three Ethical Moments in Debian*. p. 14.

[28] SRIVASTAVA, Manoj. *General Resolution: Editorial amendments to the social contract*. https://www.debian.org/vote/2004/vote_003.

[29] TOWNS, Anthony. *Social Contract GR's Affect on sarge*. https://lists.debian.org/debian-vote/2004/04/msg00074.html.

1.4.2. Contribuciones de Debian

La siguiente sección titulada: contribuiremos a la comunidad de software libre, establece que el software desarrollado dentro del proyecto debe tener una licencia que cumpla con las directrices del software libre de Debian. También se establece su política general en cuanto a la relación con otros autores cuyo software sea incluido en Debian dejando claro que dentro del proyecto se fomenta la cooperación para que las mejoras al software incluido en Debian puedan alcanzar la mayor cantidad de usuarios posible.

1.4.3. Transparencia

Respecto al manejo de errores, la política definida en la sección: No ocultaremos los problemas, establece un compromiso de transparencia entre los desarrolladores y el público en general. Aunque podría resultar conveniente en ocasiones el ocultar esta información durante el proceso de desarrollo, Debian ha optado por comprometerse a no ocultar esta información para fomentar la participación y una comunicación clara.

1.4.4. Prioridades

Dentro del contrato social se define como prioridad las necesidades de los usuarios y de la comunidad del software libre. Explícitamente el proyecto pretende brindar soporte a los usuarios para que puedan trabajar en distintos entornos de trabajo y define como uno de sus objetivos específicos, el entregar un sistema integrado de alta calidad.

1.4.5. Soporte a software no libre

En la última sección del contrato social se establece la política a seguir cuando un usuario o desarrollador requiere utilizar un software que no cumple a cabalidad con las directrices del software libre de Debian. En estos casos se puede utilizar la infraestructura provista por el proyecto Debian para la distribución de dicho software, pero se deja claro que el mismo no se considera parte del sistema operativo Debian GNU/Linux desde ningún punto de vista.

1.5. Directrices del software libre de Debian

Las directrices de software libre de Debian documentan de forma clara todos los requisitos que la licencia de un paquete de software, la documentación o cualquier tipo de archivo debe cumplir para que sea integrado al sistema operativo Debian GNU/Linux de manera oficial.

> Luego de su creación, las directrices del software libre de Debian fueron utilizadas por la empresa Sun Microsystems para establecer los criterios que debían cumplir los proyectos de software libre que deseaban usar la infraestructura de la empresa para la distribución de sus archivos[30]. En 1998 la organización llamada: Iniciativa para el código abierto, utilizó como base las directrices del software libre de Debian como base para definir los requisitos que un software debe llenar para ser considerado software de código abierto[31].

> El documento establece las directrices que son utilizadas de forma pragmática por los voluntarios del proyecto para determinar si la licencia de un software que se desea integrar a Debian GNU/Linux es software libre. Si se encuentra que no cumple con alguna de las directrices, entonces el software no puede integrarse de forma oficial. Es común que los voluntarios interesados en adicionar el software a Debian GNU/Linux contacten a los autores originales del software para solicitar cambios en la licencia para cumplir con las directrices del software libre argumentando que esto mejorará la certeza jurídica de los usuarios del programa[32].

[30] COLEMAN, Gabriella. *Debian History Roundtable Discussion*. https://gabriellacoleman.org/debian-history-roundtable-discussion/.

[31] OSI. *The Open Source Definition*. https://opensource.org/docs/osd.

[32] COLEMAN, E. Gabriella, *Three Ethical Moments in Debian.* p. 18.

Basadas en su enfoque las directrices del software libre de Debian se pueden dividir en cuatro grupos que serán tratados a continuación.

1.5.1. Distribución

Las directrices uno y dos abordan el tema de la distribución del software por personas distintas al autor, exigiendo que terceros tengan la potestad de vender o compartir de forma gratuita copias del software y además tener disponible el código fuente completo del software. Además, se requiere que la licencia del software les dé la potestad a los usuarios de distribuir el software en forma binaria o como código fuente.

1.5.2. Trabajos derivados

Las directrices tres y cuatro especifican que el usuario debe tener la potestad de hacer modificaciones al software y tener la libertad de distribuir las modificaciones si así lo desea. Esto es una práctica común dentro del proyecto Debian, pues los voluntarios a cargo del proyecto solucionan errores reportados por los usuarios, distribuyen las nuevas versiones a los usuarios de Debian y al mismo tiempo envían los cambios a los desarrolladores originales del software.

1.5.3. Discriminación

> La discriminación hacía cualquier grupo de personas está prohibida por la directriz número cinco y las directrices número seis y siete establecen que el usuario tiene la libertad de utilizar el software sin importar su propósito, ya sea comercial, científico o religioso y todos los usuarios que reciban el software deben recibir los mismos derechos[33].

[33] SPI. *The Debian Free Software Guidelines*. https://www.debian.org/social_contract.en.html#guidelines.

1.5.4. Requisitos específicos

Las directrices número ocho y nueve establecen que la licencia del software no puede imponer condiciones especiales como imponer restricciones sobre otros programas de software por el mero hecho de ser distribuidos en conjunto o dar derechos exclusivamente a los usuarios del sistema operativo Debian GNU/Linux.

1.6. Manual de normativa de Debian

> El manual de normativa de Debian define los requerimientos técnicos que cada paquete de software debe satisfacer para ser incluido en el sistema operativo Debian GNU/Linux. Los requerimientos ayudan a reducir el número de decisiones que el mantenedor de un paquete tiene que tomar y asegura que los paquetes han sido construidos siguiendo los requerimientos y van a ser compatibles con el resto del sistema operativo y las herramientas que son utilizadas para su administración[34].

La normativa de Debian específica que la información más importante de un paquete de software el sistema Debian GNU/Linux debe ser indicada en un archivo de texto con el nombre: control. En este archivo el encargado del paquete declara la relación que este paquete tiene con otros paquetes de software, por ejemplo, cuando se tiene una dependencia al momento de la instalación o un conflicto por el que no se debe instalar en combinación con otro paquete de software. Esta información es usada por el administrador de paquetes para instalar o remover dependencias de forma automática cuando un usuario desea instalar o remover un paquete de software.

La normativa Debian también especifica el proceso que el sistema debe seguir para la instalación de un paquete, define lo que un paquete puede y no puede hacer durante su configuración y el proceso para removerlo del sistema.

[34] SPI. *Debian Policy Manual*. https://www.debian.org/doc/debian-policy/index.html.

El proyecto Debian GNU/Linux recomienda el uso de librerías compartidas que son cargadas en tiempo de ejecución y especifica en la normativa de Debian la forma en que deben instalarse en el sistema de archivos del sistema. Las librerías compartidas permiten hacer llegar cualquier actualización de manera simultánea a todos los paquetes de software que dependen de la librería con solamente actualizar los archivos del paquete de software que contiene la librería.

2. FACTORES QUE AFECTAN LA PRODUCTIVIDAD EN EL PROYECTO DEBIAN GNU/LINUX

2.1. Productividad

"Un producto es todo aquello que puede ofrecerse a un mercado para satisfacer una necesidad o una demanda y producción es el proceso mediante el cual se transforman diversos insumos materiales e inmateriales en productos, ya sean bienes o servicios"[35].

La productividad es la medida de la eficiencia de la producción, es decir la cantidad de productos que se obtienen comparada contra la cantidad de insumos que se utilizaron para dicha producción. Por ejemplo, en la industria de la manufactura es común el uso de la cantidad de unidades producidas por unidad de tiempo invertido, como una medida de la productividad. Sin embargo, en las organizaciones que producen bienes intangibles como investigaciones científicas o software, la productividad es más difícil de medir y debido a esto, se deben tomar en cuenta factores específicos de las organizaciones bajo análisis[36].

En el caso de Debian GNU/Linux, el producto final es un sistema operativo que está compuesto por miles de paquetes de software individuales. El equipo de desarrollo de Debian GNU/Linux es responsable por la creación y mantenimiento de cada uno de estos paquetes de software. Aunque la mayoría de estos paquetes contienen software desarrollado por personas fuera del

[35] KOTLER, Philip. *Principles of marketing*. p. 5.

[36] SADOWSKI, Caitlin y ZIMMERMANN, Thomas. *Rethinking Productivity in Software Engineering*. p. 49-55.

proyecto, cada uno de los desarrolladores de Debian GNU/Linux es responsable de probar, modificar y actualizar el software para que se adapte a las necesidades de los usuarios del sistema operativo Debian GNU/Linux.

Las tareas que resultan en el mayor beneficio para los usuarios del sistema operativo Debian GNU/Linux son:

Creación de paquetes de software nuevos
Actualizaciones mayores de un paquete de software existente

Ambas actividades proveen al usuario nuevas funcionalidades o mejoras significativas al software, es por esto que ambas actividades se utilizarán en el presente capítulo para la evaluación de la productividad del proyecto Debian GNU/Linux.

La otra variable que se debe tomar en cuenta para medir la productividad es el tiempo entre entregas. Se puede realizar esta simplificación gracias a que todo el trabajo es realizado por voluntarios, de manera que, hablando de forma general, cualquiera de las siguientes opciones puede proveer una mejora a la productividad del proyecto Debian GNU/Linux:

- Maximizar la cantidad de paquetes de software nuevos y actualizaciones mayores.
- Minimizar el tiempo entre entregas a los usuarios finales.

Desde el punto de vista de los usuarios de Debian GNU/Linux se suele tener la percepción que la publicación de una nueva versión del sistema operativo Debian GNU/Linux toma mucho tiempo. Esto puede ser un indicador de un problema de eficiencia que vale la pena evaluar.

En las siguientes secciones se analizarán los resultados obtenidos en conjunto por el proyecto, así como un análisis del trabajo a nivel individual, tratando de encontrar posibles mejoras, ya sea el aprovechar de mejor manera el trabajo que realiza cada desarrollador de Debian GNU/Linux o identificar períodos que puedan ser reducidos en el ciclo de trabajo de la organización.

2.2. Fuente de datos

"Los datos que se presentan en las siguientes secciones son el resultado del procesamiento de la información obtenida haciendo uso de Debian Ultimate Database, que es un repositorio centralizado de múltiples métricas del proceso de desarrollo de Debian GNU/Linux"[37]. Los datos fueron agrupados usando como delimitador principal cada una de las fechas de entrega de nuevas versiones de Debian GNU/Linux desde el año 2009 hasta el año 2019.

2.3. Antecedentes de productividad del proyecto

La productividad de un proyecto de software suele ser evaluada de forma subjetiva por sus usuarios, lo que en el caso de Debian GNU/Linux resulta en algunos casos, en una evaluación negativa debido al tiempo que históricamente ha pasado entre cada nueva versión de este sistema operativo. Los desarrolladores de Debian GNU/Linux en cambio están al tanto de toda la actividad y nuevas funcionalidades completadas de forma individual, aunque estás aún no estén disponibles al usuario final. Esto puede causar una percepción diferente entre el usuario final y el equipo de desarrollo de Debian GNU/Linux.

En esta sección se analizarán los antecedentes de productividad del proyecto Debian GNU/Linux. Para ello, como se menciona en la sección anterior

[37] SPI. *Ultimate Debian Database*. https://udd.debian.org/.

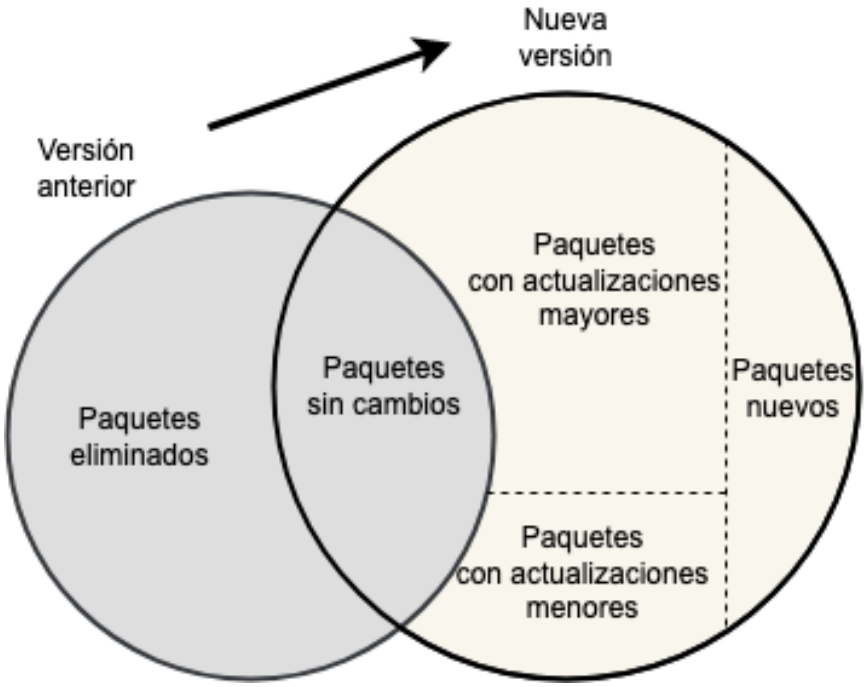
se analizará la producción de paquetes (nuevos y actualizaciones) así como la cantidad de tiempo invertido entre cada nueva versión del sistema operativo.

2.3.1. Productividad grupal

A continuación, se presentan las métricas de productividad de la organización de forma grupal, es decir los resultados acumulados sin detallar la productividad individual de cada integrante de la organización. La cantidad de paquetes de software entregados al usuario final constituyen la medida más efectiva de productividad entre cada entrega, pero cada nueva versión de Debian GNU/Linux se basa en la versión anterior, es por eso que es importante establecer una comparación entre los paquetes de software de una versión con los paquetes de software que formaron parte de la versión anterior. Esto se ilustra en la siguiente figura:

Como se observa en la figura anterior cada nueva versión de Debian GNU/Linux tiene como base el conjunto de paquetes de software de la versión anterior. Durante el desarrollo de la nueva versión muchos paquetes son eliminados debido a que presentan errores graves o su funcionalidad se considera obsoleta

Figura 6. **Relación entre diferentes versiones Debian GNU/Linux**



Fuente: elaboración propia, empleando Drawio.

También se presenta un subconjunto de paquetes a los cuales no se les aplica ningún cambio y además de estos se agregan:

- Actualizaciones menores: cambios para solucionar errores, cambios en la configuración o mejoras en la interacción con otros paquetes de software.
- Actualizaciones mayores: cambios que agregan nuevas funcionalidades al paquete de software.
- Paquetes de software nuevos: son paquetes que no formaban parte de la versión anterior de Debian GNU/Linux y son agregados para proveer nuevas funcionalidades al usuario final.

2.3.1.1. Cantidad total de paquetes

Debido a que Debian GNU/Linux está compuesto por miles de paquetes de software que proveen distintas funcionalidades a sus usuarios, es conveniente considerar la cantidad de paquetes entregados con cada versión, para tener una evaluación objetiva de la productividad grupal de la organización. Es decir que, si en cada nueva versión la cantidad de paquetes sube o baja, se puede considerar que la productividad también ha mejorado o ha empeorado.

> Al analizar los registros históricos se puede observar que la cantidad total de paquetes de software entregados en cada nueva versión de Debian GNU/Linux se ha ido incrementando, pasando de 12 099 en el año 2009, a 28 404 en el año 2019 con un crecimiento promedio de 3 031 paquetes en cada entrega. La tabla que se presenta a continuación detalla la cantidad de paquetes de software entregados en las últimas seis versiones del sistema operativo Debian GNU/Linux[38]:

Tabla I. **Paquetes de software entregados**

Fecha	Entrega	Paquetes entregados	Crecimiento
2009-02-14	5 (Lenny)	12 099	1 886
2011-02-06	6 (Squeeze)	14 585	2 486
2013-05-04	7 (Wheezy)	17 145	2 560
2015-04-26	8 (Jessie)	20 536	3 391
2017-06-17	9 (Stretch)	24 723	4 187
2019-07-06	10 (Buster)	28 404	3 681

Fuente: elaboración propia, empleando Microsoft Word.

En la tabla anterior se observan las dos métricas principales:

[38] SPI. *Ultimate Debian Database*. https://udd.debian.org/.

Paquetes entregados: es el número total de paquetes incluidos en cada versión. Crecimiento: corresponde al crecimiento en la cantidad de paquetes con respecto a la versión anterior. Por ejemplo, la versión seis de Debian GNU/Linux presento un crecimiento de 2 486 paquetes de software al compararla con el total de paquetes de la versión cinco.

El detalle más importante en la tabla anterior es que desde el 2009, el total de paquetes entregados con cada versión de Debian GNU/Linux se ha ido incrementando de forma consistente.

2.3.1.2. Paquetes no modificados y eliminados

Cada nueva versión de Debian GNU/Linux está basada en el conjunto de paquetes de software que formaron parte de la versión anterior, debido a esto existe un conjunto de paquetes que pasan a la siguiente versión sin ningún cambio. Es importante identificar dichos paquetes porque podrían sesgar la percepción de productividad, cuando en realidad estos paquetes no requirieron ningún tipo de trabajo por parte del equipo de desarrollo.

Existe también otro conjunto de paquetes de software que es eliminado de una versión, al presentar problemas graves o al considerarse obsoletos. Es importante conocer la cantidad de paquetes eliminados porque si esta fuera muy alta tendría un impacto significativo en la cantidad total de paquetes en cada nueva versión.

En la siguiente tabla se presenta el detalle de paquetes mantenidos y eliminados en cada versión de Debian GNU/Linux desde el 2009. Por ejemplo, en la última entrega 7 266 paquetes se mantuvieron en la misma versión y se eliminaron 1 822 paquetes de software.

Tabla II. **Paquetes mantenidos y eliminados**

Fecha	Entrega	Misma versión	Eliminados
2009-02-14	5 (Lenny)	1 608	1 270
2011-02-06	6 (Squeeze)	2 542	1 606
2013-05-04	7 (Wheezy)	3 543	1 786
2015-04-26	8 (Jessie)	4 937	1 361
2017-06-17	9 (Stretch)	5 262	1 757
2019-07-06	10 (Buster)	7 266	1 822

Fuente: elaboración propia, empleando Microsoft Word.

En la tabla anterior se pueden observar las columnas tituladas:

- Misma versión: que indica la cantidad de paquetes que no sufrieron ningún cambio con respecto a la versión anterior del sistema operativo.
- Eliminados: que indica la cantidad de paquetes que fueron eliminados previo a completar la entrega de la nueva versión del sistema operativo.

Así también, se observa que, aunque hay un número importante de paquetes de software que se mantienen entre versiones, este número no constituye la mayoría de los paquetes de software. También se puede observar que, aunque más de mil paquetes son borrados entre cada entrega, esto no ha sido un obstáculo para que la cantidad total de paquetes aumente con cada nueva versión Debian GNU/Linux.

2.3.1.3. Paquetes actualizados y paquetes nuevos

El producto principal del trabajo de los desarrolladores está conformado por tres categorías: las actualizaciones menores, que pueden consistir en cambios para solucionar errores, cambios en la configuración, o mejoras en la interacción con otros paquetes; las utilerías del sistema; actualizaciones mayores, las cuales consisten en la inclusión de nuevas funcionalidades; y por último, los paquetes de software totalmente nuevos. La última entrega realizada incluyó 5 492 paquetes nuevos, 10 480 actualizaciones mayores y 5 166 actualizaciones menores. En la siguiente tabla se detallan los resultados para cada entrega desde el 2009 a la fecha:

Tabla III. **Actualizaciones y nuevos paquetes de software**

Fecha	Entrega	Actualización menor	Actualización mayor	Nuevos
2009-02-14	5 (Lenny)	2 815	4 520	3 156
2011-02-06	6 (Squeeze)	2 578	5 366	4 099
2013-05-04	7 (Wheezy)	3 088	6 185	4 329
2015-04-26	8 (Jessie)	3 266	7 589	4 744
2017-06-17	9 (Stretch)	4 495	9 034	5 932
2019-07-06	10 (Buster)	5 166	10 480	5 492

Fuente: elaboración propia, empleando Microsoft Word.

Se puede observar que de forma consistente tanto la cantidad de paquetes nuevos y la cantidad de actualizaciones mayores superó a la cantidad de actualizaciones menores. Esto es un resultado positivo debido a que esta es la forma en que los usuarios finales obtienen nuevas funcionalidades.

2.3.1.4. Porcentaje de cambio

Debido a que las actualizaciones mayores y los paquetes nuevos proveen los mayores beneficios para los usuarios, a continuación, se evalúa el porcentaje de estos con relación a la cantidad total de paquetes de software incluidos en la versión anterior.

Luego de analizar los registros históricos se puede observar que en promedio, desde el año 2009, el 43,61 % de paquetes reciben una actualización mayor en cada nueva entrega y se introduce un 28,87 % de paquetes nuevos. En la siguiente tabla se detallan los porcentajes de cambio en cada entrega analizada:

Tabla IV. **Cambio porcentual con respecto a la entrega anterior**

Versión	Total, anterior	Actualización mayor	Paquetes nuevos
5 (Lenny)	10 213	44,26 %	30,90 %
6 (Squeeze)	12 099	44,35 %	33,88 %
7 (Wheezy)	14 585	42,41 %	29,68 %
8 (Jessie)	17 145	44,26 %	27,67 %
9 (Stretch)	20 536	43,99 %	28,89 %
10 (Buster)	24 723	42,39 %	22,21 %

Fuente: elaboración propia, empleando Microsoft Word.

En la tabla anterior se presentan las columnas tituladas:

Versión: indica la versión de Debian GNU/Linux que se está analizando.

Total, anterior: indica la cantidad total de paquetes de software que fueron entregados en la versión anterior de Debian GNU/Linux. Esta cantidad es usada como referencia para calcular las siguientes variables.

Actualización mayor: indica el porcentaje de paquetes que recibieron una actualización mayor. Este porcentaje se calculó usando como referencia la cantidad total de paquetes entregados en la versión anterior de Debian GNU/Linux.

Paquetes nuevos: indica el porcentaje de paquetes nuevos incluidos en cada versión analizada. El porcentaje fue calculado utilizando como referencia la cantidad total de paquetes entregados en la versión anterior de Debian GNU/Linux debido a que eso permite comparar de mejor manera el cambio entre versiones.

El detalle más importante que se presenta en la tabla anterior es que el porcentaje de cambio entre las distintas versiones analizadas es similar, y de ser conveniente, puede ser utilizado como una métrica del avance en el desarrollo de futuras versiones del sistema operativo. Por ejemplo: es posible elaborar un reporte mensual con el porcentaje de paquetes que han recibido una actualización mayor y utilizar dicha métrica como uno de los criterios para decidir el mejor punto en el tiempo para realizar la siguiente entrega del sistema operativo.

2.3.1.5. Tiempo invertido en cada entrega

Como se detalló en anteriormente, cada nueva versión de Debian GNU/Linux provee al usuario final nuevas funcionalidades en la forma de actualizaciones y nuevos paquetes de software, sin embargo, para lograr este

objetivo el equipo de desarrollo del proyecto Debian debe invertir una cantidad considerable de tiempo. A continuación, se analiza el tiempo que históricamente le ha tomado al equipo de desarrollo de Debian GNU/Linux la creación las últimas seis versiones.

En relación al tiempo invertido por cada entrega desde el 2009, se tiene un promedio de 24 meses. El peor de los casos se observó con la versión número siete, la cual requirió 26 meses de trabajo, sin embargo, al compararlo con el promedio se observa un retraso de dos meses, lo cual no es significativo. En la siguiente tabla se detalla el tiempo que tomo la elaboración de cada versión de Debian GNU/Linux en los últimos diez años:

Tabla V. **Tiempo entre cada entrega**

Entrega	**Meses**
5 (Lenny)	22
6 (Squeeze)	23
7 (Wheezy)	26
8 (Jessie)	23
9 (Stretch)	25
10 (Buster)	24

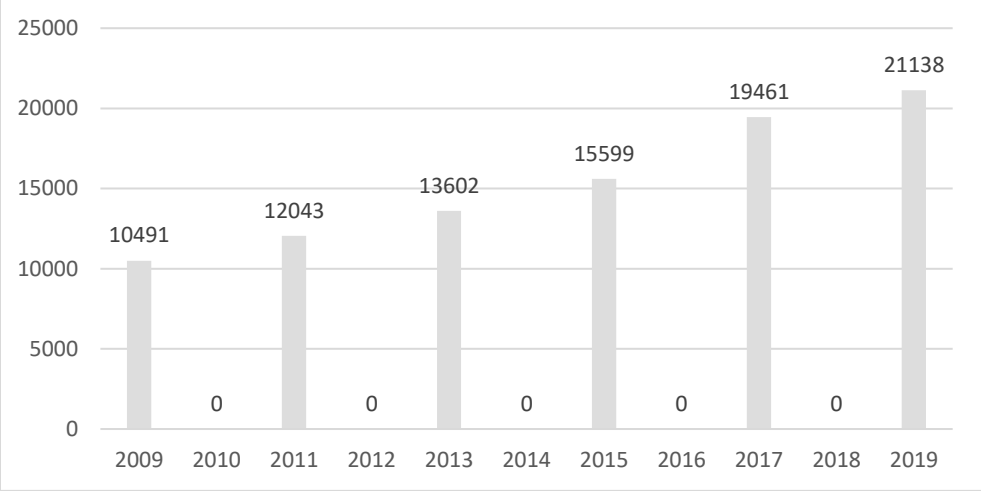
Fuente: elaboración propia, empleando Microsoft Word.

La tabla anterior muestra que el tiempo de desarrollo no se ha incrementado de forma significativa, a pesar de que con cada entrega se ha incrementado la cantidad de paquetes de software, esto puede considerarse un resultado positivo porque la constancia en los tiempos de entrega permite cumplir con las expectativas de los usuarios finales, sin embargo, dos años es un período

significativo y es conveniente buscar optimizar el tiempo de desarrollo para entregar nuevas funcionalidades a los usuarios finales con una inversión menor de tiempo.

Para cada entrega se inicia un proceso de coordinación, para la publicación de todos los paquetes de software, y la notificación al público en general de su disponibilidad. Aunque el resultado del trabajo en grupo provee gran cantidad de paquetes nuevos y actualizaciones, el desarrollador solo ve completado el ciclo con la entrega final cada dos años, pues como se ilustra en la siguiente figura durante los años 2010, 2012, 2014, 2016 y 2018 no se realizaron nuevas entregas[39]:

Figura 7. **Entrega de paquetes nuevos y actualizados**



Fuente: elaboración propia, empleando Microsoft Word.

En síntesis, la productividad fruto del trabajo en grupo del equipo de desarrollo del proyecto Debian GNU/Linux se puede resumir de la siguiente manera: en promedio desde el 2009 el 43,61 % de paquetes reciben una actualización mayor en cada nueva entrega de Debian GNU/Linux y un promedio de 28.87 % de paquetes nuevos son agregados, para ello el proyecto invierte un

[39] SPI. *Debian Releases*. https://www.debian.org/releases/.

promedio de 24 meses en cada ciclo de trabajo. Es decir, que aunque se invierte una cantidad similar de tiempo en cada nueva versión, cada una incluye más y más paquetes de software. Esto indica que se puede mejorar el balance entre la entrega de nuevas funcionalidades con la cantidad de tiempo invertido en el desarrollo de una nueva versión. Para el usuario final no solamente es valioso el recibir una mayor cantidad de nuevas funcionalidades sino también el tener acceso a las mismas, en el menor tiempo posible.

2.3.2. Productividad individual

El proyecto Debian GNU/Linux no lleva un registro de las horas de trabajo de cada uno de los voluntarios, sin embargo, se tiene un registro detallado de forma individual de cada paquete de software y sus respectivas versiones durante el ciclo de desarrollo. Gracias a que cada paquete de software tiene asignado por lo menos a un miembro del equipo de desarrollo, esto permite realizar un análisis de la productividad de forma individual, midiendo la cantidad de paquetes producidos o actualizados y el punto en el tiempo en que fueron completados por sus respectivos desarrolladores.

2.3.2.1. Actualizaciones mayores

La primera métrica por analizar desde el punto de vista individual es la cantidad de actualizaciones mayores, las cuales consisten en cambios que agregan nuevas funcionalidades a un paquete de software existente.

La entrega final al usuario del sistema operativo Debian GNU/Linux solamente incluye la última versión de cada paquete de software, sin embargo, la productividad individual suele ser superior, porque es muy común que los desarrolladores completen múltiples actualizaciones mayores durante el mismo

ciclo de desarrollo de una nueva versión de Debian GNU/Linux. Cada actualización completada está lista para ser entregada al usuario final, sin embargo, se debe esperar a que se coordine una nueva entrega del sistema operativo completo, lo cual, como se mencionó en secciones anteriores ocurre en promedio cada veinticuatro meses.

Como ejemplo de la productividad individual se presenta a continuación el caso del paquete del núcleo Linux: entre junio de 2017 y febrero de 2019 se realizaron treinta y cuatro entregas mayores, de las cuales treinta y tres nunca fueron entregadas al usuario final. Como se puede observar en la siguiente tabla, la última versión que fue incluida fue completada en febrero de 2019, cinco meses antes de la entrega final:

Tabla VI. **Actualizaciones mayores del paquete Linux**

Año-mes	Actualización mayor	Año-mes	Actualización mayor
2017-06	4.11.6	2018-05	4.16.12
2017-07	4.11.11	2018-06	4.16.16
2017-08	4.12.6	2018-07	4.17.3
2017-09	4.12.12	2018-07	4.17.6
2017-09	4.12.13	2018-07	4.17.8
2017-10	4.13.4	2018-08	4.17.14
2017-10	4.13.10	2018-08	4.17.15
2017-11	4.13.13	2018-08	4.17.17
2017-12	4.14.2	2018-09	4.18.6
2017-12	4.14.7	2018-09	4.18.8
2018-01	4.14.12	2018-10	4.18.10
2018-01	4.14.13	2018-11	4.18.20
2018-02	4.14.17	2018-12	4.19.9
2018-02	4.15.4	2018-12	4.19.12
2018-03	4.15.11	2018-12	4.19.13
2018-04	4.15.17	2019-01	4.19.16
2018-05	4.16.5	2019-02	4.19.20

Fuente: elaboración propia, empleando Microsoft Word.

También se puede observar que entre junio de 2017 y julio de 2019 los desarrolladores a cargo del paquete de software Ansible, completaron dieciocho actualizaciones mayores de las cuales solamente la última, elaborada en febrero de 2019 fue entregada al usuario final. El detalle de cada una de las

actualizaciones mayores del paquete de software Ansible se presenta en la tabla a continuación:

Tabla VII. **Actualizaciones mayores del paquete Ansible**

Año-mes	Actualización mayor	Año-mes	Actualización Mayor
2017-08	2.3.1.0	2018-06	2.5.5
2017-09	2.3.2.0	2018-07	2.6.1
2017-10	2.4.0.0	2018-08	2.6.3
2017-12	2.4.2.0	2018-09	2.6.4
2018-02	2.4.3.0	2018-09	2.6.5
2018-03	2.5.0	2018-11	2.7.1
2018-04	2.5.1	2018-12	2.7.5
2018-05	2.5.2	2019-01	2.7.6
2018-05	2.5.3	2019-02	2.7.7

Fuente: elaboración propia, empleando Microsoft Word.

Los casos presentados anteriormente no son los únicos, por lo que a continuación se presenta un análisis general de cada uno de los ciclos de desarrollo de Debian GNU/Linux desde el 2007 y como han ocurrido múltiples instancias de este problema en el que no se aprovecha al máximo el producto del trabajo de los desarrolladores.

Al analizar los datos correspondientes al período de desarrollo de la versión cinco de Debian GNU/Linux que duró de abril de 2007 a febrero de 2009, se puede observar que 4 680 paquetes de software tuvieron por lo menos una

actualización mayor que no fue entregada al usuario final, lo que constituye el 38 % de los paquetes entregados en esa versión.

A continuación, se presentan los siguientes indicadores de actualizaciones mayores no entregadas en los períodos 2007 – 2009, 2009 – 2011, 2011 – 2013, 2013 – 2015, 2015 – 2017, 2017 – 2019:

Paquetes afectados: es la cantidad de paquetes de software que tuvieron por lo menos una actualización mayor que no fue entregada al usuario final. Esta cantidad solo toma en cuenta paquetes de software únicos, sin contar las ocasiones en las que un paquete de software recibió múltiples actualizaciones mayores.

Porcentaje del total: es el porcentaje de paquetes de software afectados respecto al total de paquetes entregados al final del ciclo de desarrollo.

Promedio de actualizaciones por paquete: representa el promedio de actualizaciones mayores que fueron completadas para un mismo paquete de software y que no fueron entregadas al usuario final.

Total de actualizaciones mayores no entregadas: representa la cantidad total de actualizaciones mayores que no fueron entregadas al usuario final. Esta cantidad toma en cuenta las múltiples actualizaciones que se completaron para un mismo paquete de software.

Tabla VIII. **Actualizaciones mayores no entregadas 2007 – 2009**

Indicador	**Cantidad**
Paquetes afectados	4 618
Porcentaje del total	38 %
Promedio de actualizaciones por paquete	3,33
Total de actualizaciones mayores no entregadas	15 357

Fuente: elaboración propia, empleando Microsoft Word.

Como se detalla en la siguiente tabla, durante el siguiente período de desarrollo que duro de febrero de 2009 a febrero de 2011, se observaron 5 154 paquetes afectados. En promedio, los paquetes afectados recibieron tres actualizaciones mayores que no fueron entregadas al usuario final.

Tabla IX. **Actualizaciones mayores no entregadas 2009 – 2011**

Indicador	**Cantidad**
Paquetes afectados	5 154
Porcentaje del total	35 %
Promedio de actualizaciones por paquete	3,21
Total de actualizaciones mayores	16 519

Fuente: elaboración propia, empleando Microsoft Word.

En el período de desarrollo de la versión siete de Debian GNU/Linux se observa que el porcentaje de paquetes afectados bajó levemente a 32 % como se detalla en la siguiente tabla:

Tabla X. **Actualizaciones mayores no entregadas 2011 – 2013**

Indicador	**Cantidad**
Paquetes afectados	5 496
Porcentaje del total	32 %
Promedio de actualizaciones por paquete	2,82
Total de actualizaciones mayores	15 517

Fuente: elaboración propia, empleando Microsoft Word.

En el siguiente período de desarrollo que abarcó de mayo de 2013 a abril de 2015 un total de 6 142 paquetes tuvieron en promedio 2,74 actualizaciones mayores que no fueron entregados al usuario final.

Tabla XI. **Actualizaciones mayores no entregadas 2013 – 2015**

Indicador	**Cantidad**
Paquetes afectados	6 142
Porcentaje del total	30 %
Promedio de actualizaciones por paquete	2,74
Total de actualizaciones mayores	16 837

Fuente: elaboración propia, empleando Microsoft Word.

El porcentaje de paquetes afectados llego a 34 % durante el desarrollo de la versión diez de Debian GNU/Linux y como se detalla en la siguiente tabla tuvo un promedio de 3,12 actualizaciones no entregadas entre los paquetes afectados lo cual es bastante cercano al promedio del período anterior:

Tabla XII. **Actualizaciones mayores no entregadas 2015 – 2017**

Indicador	Cantidad
Paquetes afectados	8 378
Porcentaje del total	34 %
Promedio de actualizaciones por paquete	3,12
Total de actualizaciones mayores	26 145

Fuente: elaboración propia, empleando Microsoft Word.

La cantidad de paquetes afectados durante el período de desarrollo de junio de 2017 a julio de 2019 llegó al 31 % del total de paquetes que formaron parte de la última entrega oficial de Debian GNU/Linux. Como se puede observar en la siguiente tabla, en total se tuvieron 25 612 actualizaciones mayores que no fueron entregadas al usuario final:

Tabla XIII. **Actualizaciones mayores no entregadas 2017 – 2019**

Indicador	Cantidad
Paquetes afectados	8 794
Porcentaje del total	31 %
Promedio de actualizaciones por paquete	2,91
Total de actualizaciones mayores	25 612

Fuente: elaboración propia, empleando Microsoft Word.

Como se observa en las tablas de datos anteriores en todos los ciclos de desarrollo desde el año 2007 ha existido entre un 30 % y 38 % de paquetes de software en los que la productividad del equipo de desarrollo no es aprovechada

al máximo, este es un recurso valioso puesto que un cambio en la gestión de entregas de software del proyecto puede aprovecharlo para incrementar la productividad grupal beneficiando al usuario final con más funcionalidades en un menor tiempo.

2.3.2.2. Tiempo invertido del ciclo de desarrollo

A continuación, se analiza el tiempo invertido durante el ciclo de desarrollo, utilizando como referencia:

- La primera actualización mayor completada de cada paquete de software.
- Paquetes nuevos completados durante el ciclo de desarrollo.

El identificar el punto en el tiempo de este tipo de eventos permitirá tener una mejor comprensión del ritmo de producción individual del equipo de trabajo durante el ciclo de desarrollo.

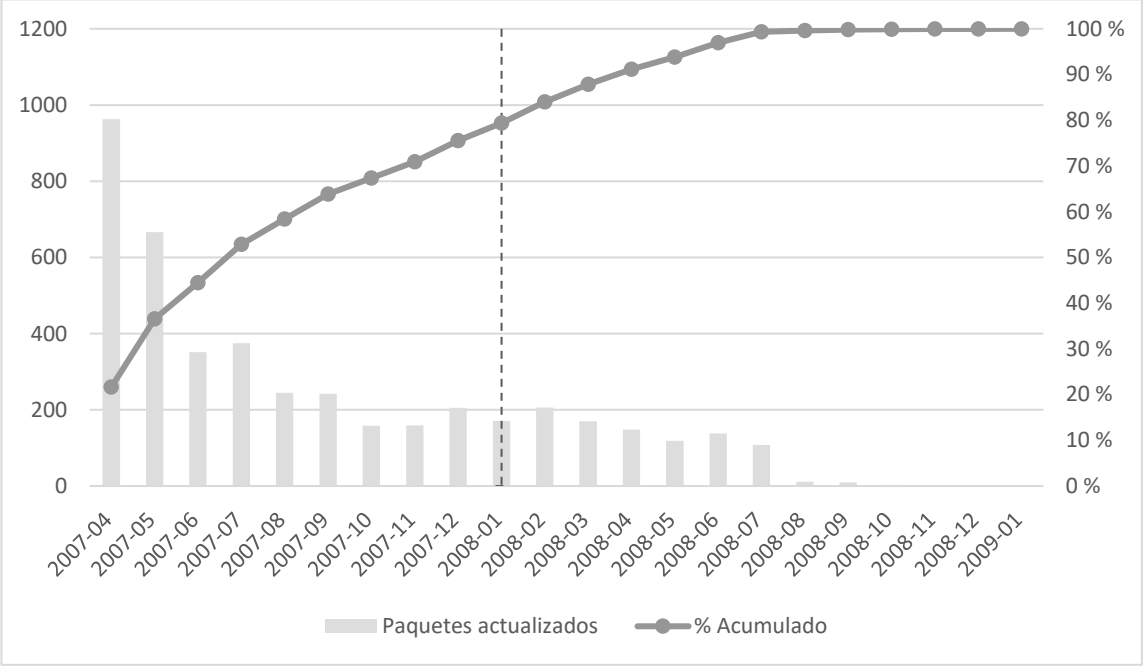
Es importante remarcar que en los últimos meses de todos los ciclos de desarrollo, luego de coordinar vía correo electrónico, todo el equipo detiene la creación de nuevos paquetes y actualizaciones mayores para enfocarse exclusivamente en la resolución de errores hasta completar la siguiente entrega final del sistema operativo completo. Lo anterior se ve reflejado en todas las figuras que se presentan a continuación. Para cada una se consideran los siguientes elementos:

- Eje horizontal: indica el año y mes correspondiente.
- Barras verticales: representa la cantidad de paquetes que recibieron su primera actualización mayor durante el mes.

- Línea del porcentaje: indica el porcentaje acumulado de paquetes que ya han recibido su primera actualización mayor. Es decir, el porcentaje del trabajo total que se completó en ese mes.

- Línea vertical discontinua: se utiliza una línea vertical discontinua para marcar el mes en el que el 80 % del trabajo ya se ha completado.

Figura 8. **Primera actualización mayor 2007 - 2009**



Fuente: elaboración propia, empleando Microsoft Word.

En la figura anterior se ilustra cómo se van acumulando las actualizaciones mayores para distintos paquetes (sin contar las actualizaciones que pertenecen a un mismo paquete) y se observa que aproximadamente el 80 % de los paquetes que recibieron actualizaciones mayores ya habían sido completados por los

desarrolladores durante los primeros diez meses de los veintidós que duro el ciclo de desarrollo de Debian GNU/Linux versión cinco.

A continuación, se puede observar la gráfica correspondiente a los paquetes nuevos, la cual presenta los siguientes elementos:

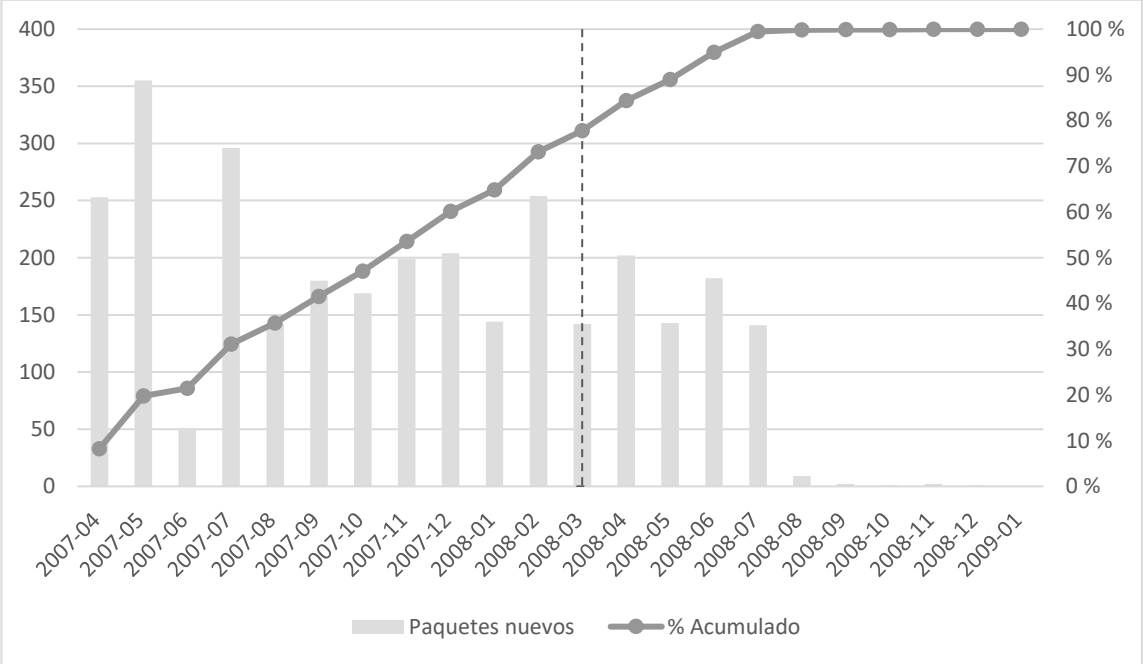
- Eje horizontal: indica el año y mes correspondiente.

- Barras verticales: representa la cantidad de paquetes nuevos completados durante el mes.

- Línea del porcentaje: Indica el porcentaje acumulado de paquetes nuevos, es decir, el porcentaje del trabajo total que se completó en ese mes.

- Línea vertical discontinua: se utiliza una línea vertical discontinua para marcar el mes en el que el 80 % del trabajo ya se ha completado.

Figura 9. **Paquetes nuevos 2007 - 2009**

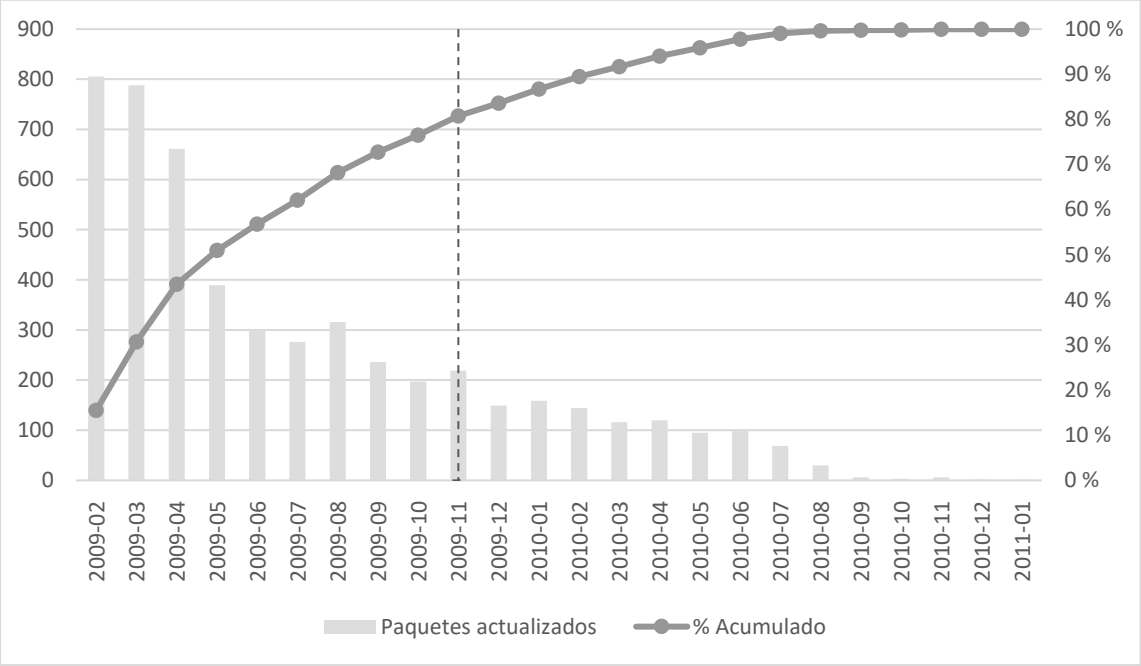


Fuente: elaboración propia, empleando Microsoft Word.

La gráfica anterior muestra una productividad uniforme que alcanzó a producir el 80 % de los paquetes nuevos en los primeros 12 meses del ciclo de desarrollo.

Los primeros tres meses del siguiente ciclo de desarrollo fueron particularmente productivos, durante estos tres meses 2 254 paquetes completaron una primera actualización mayor y como se ilustra a continuación el 80 % de primeras actualizaciones mayores ya se habían completado al cumplirse el décimo mes de desarrollo:

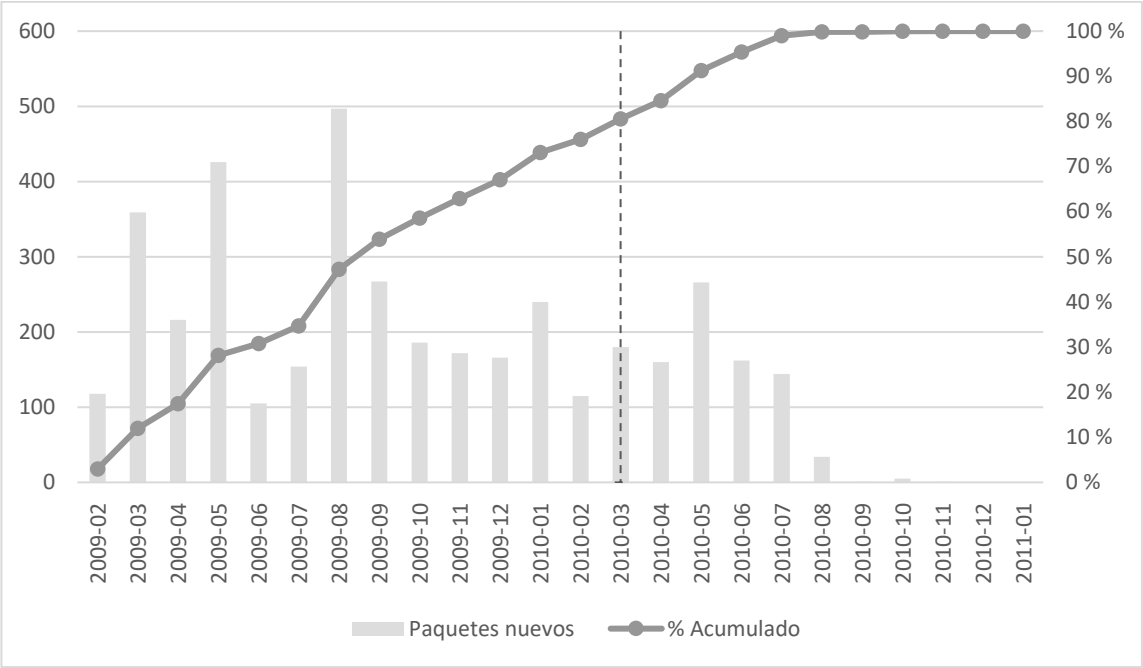
Figura 10. **Primera actualización mayor 2009 - 2011**



Fuente: elaboración propia, empleando Microsoft Word.

La cantidad de paquetes nuevos que fueron creados durante agosto de 2009 fue superior al resto de los meses de este ciclo, sin embargo, la creación de paquetes nuevos mostro un avance regular, como se muestra en la siguiente figura:

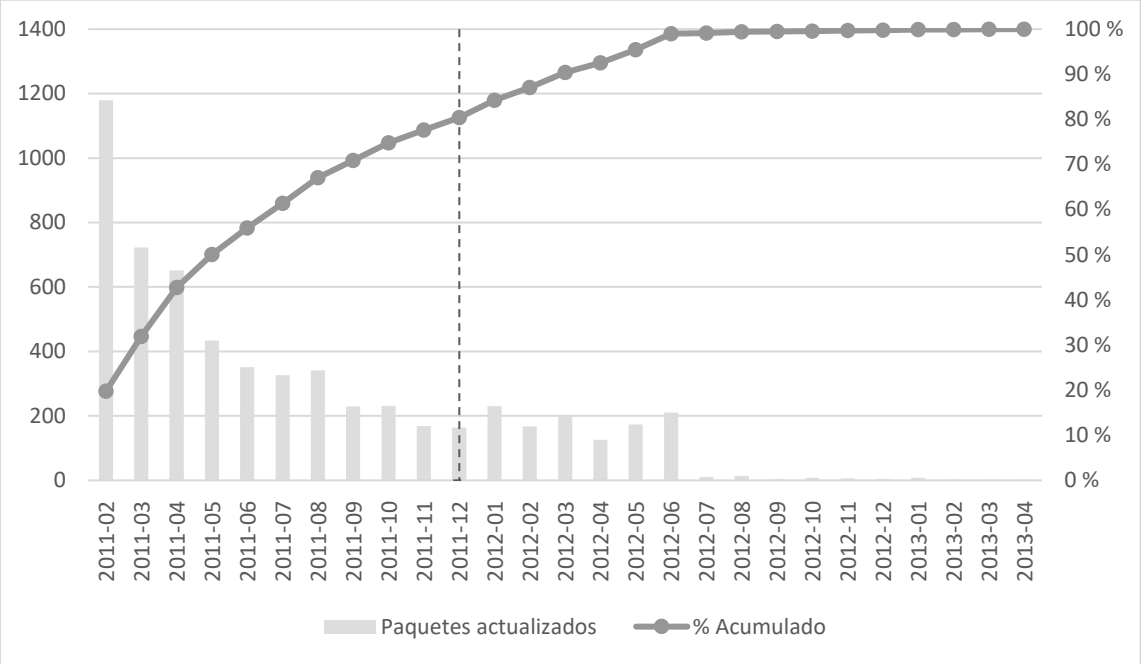
Figura 11. **Paquetes nuevos 2009 - 2011**



Fuente: elaboración propia, empleando Microsoft Word.

El período de desarrollo de 2011 a 2013 duró veintiséis meses y fue el período más largo de los que se analizan, a pesar de esto aproximadamente el 80 % de los paquetes que recibieron actualizaciones mayores ya habían recibido su primera actualización por parte de los desarrolladores en solo once meses, es decir antes de alcanzar la mitad del ciclo de desarrollo. Esto se detalla en la figura que se presenta a continuación:

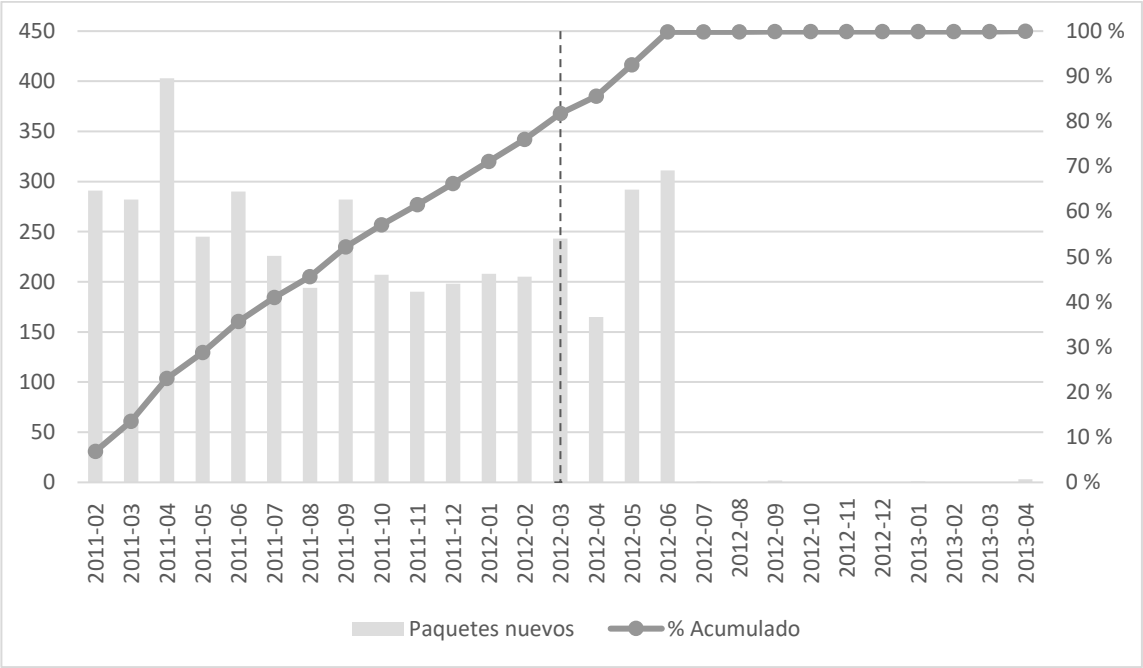
Figura 12. **Primera actualización mayor 2011 - 2013**



Fuente: elaboración propia, empleando Microsoft Word.

La cantidad de paquetes nuevos que fueron creados mensualmente mantuvo un ritmo estable durante el ciclo de desarrollo de 2011 a 2013. El 80 % de paquetes nuevos fueron completados en el mes número catorce de los veintiséis meses que duró el ciclo de desarrollo de 2011 a 2013.

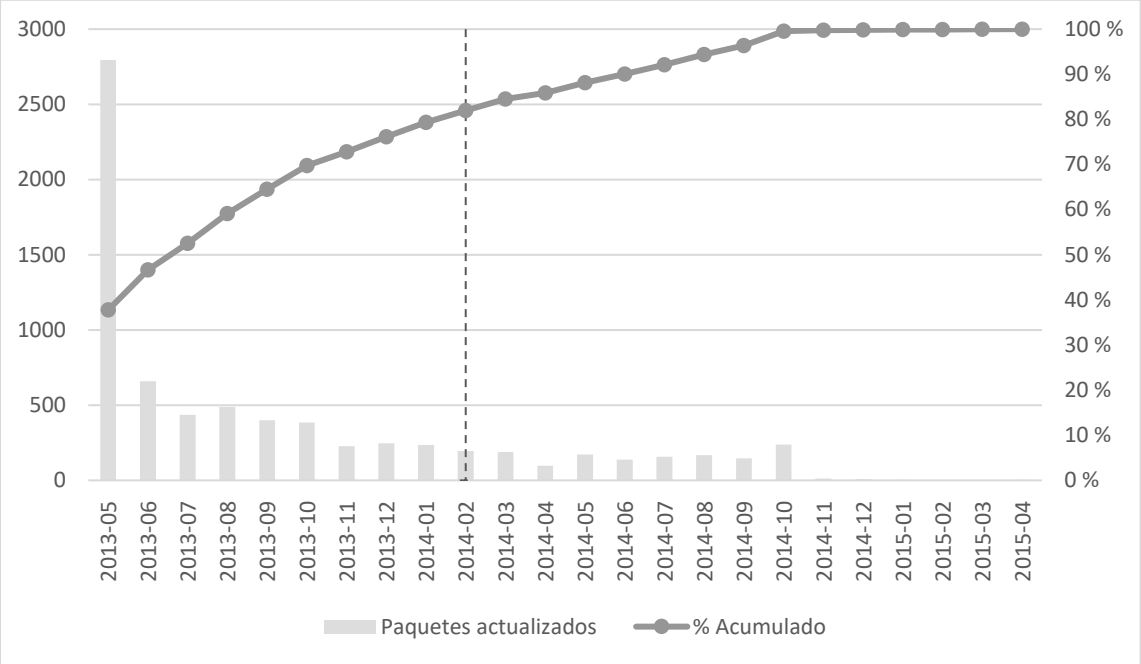
Figura 13. **Paquetes nuevos 2011 - 2013**



Fuente: elaboración propia, empleando Microsoft Word.

El período ciclo de desarrollo anterior fue el período más largo de los que se analizan en la presente investigación esto tuvo como consecuencia que muchas de las actualizaciones mayores para el siguiente ciclo se completaran en el primer mes. Esto se evidencia en la siguiente figura. También se observa que el 80 % de los paquetes que recibieron una actualización ya habían sido completados en el mes número diez de los 23 que duro este ciclo de desarrollo.

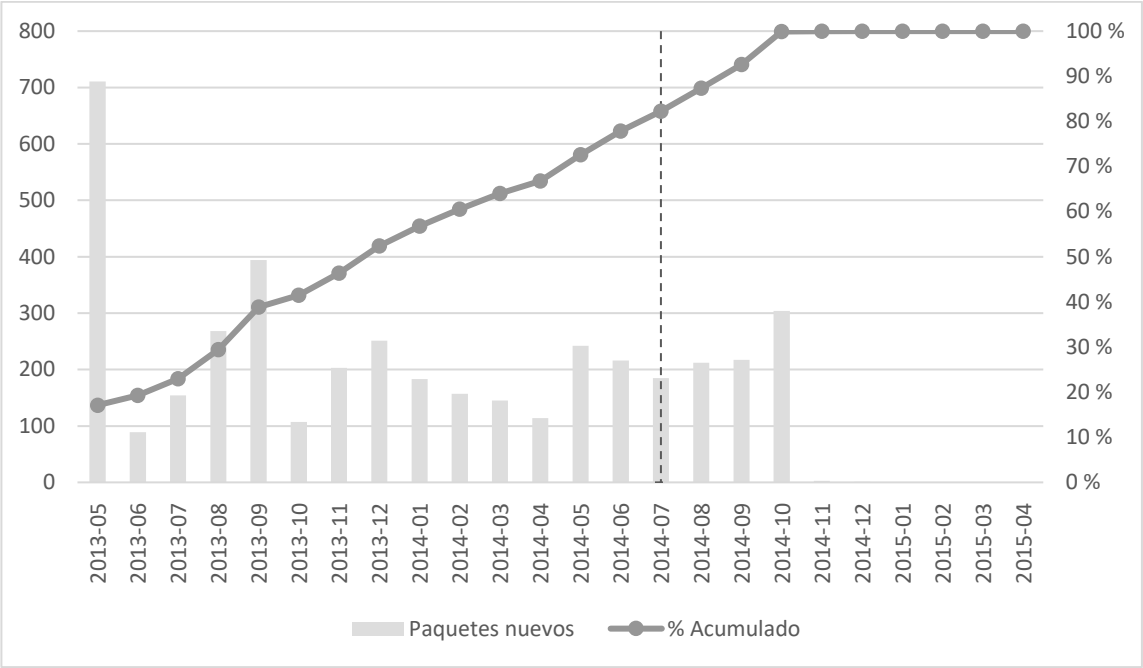
Figura 14. **Primera actualización mayor 2013 - 2015**



Fuente: elaboración propia, empleando Microsoft Word.

La figura correspondiente a la creación mensual de paquetes nuevos también refleja un primer mes en el que ya se tienen disponibles 711 paquetes nuevos completados por el equipo de desarrollo Debian GNU/Linux, mientras que el 80 % de los paquetes nuevos es completado al final del mes número quince.

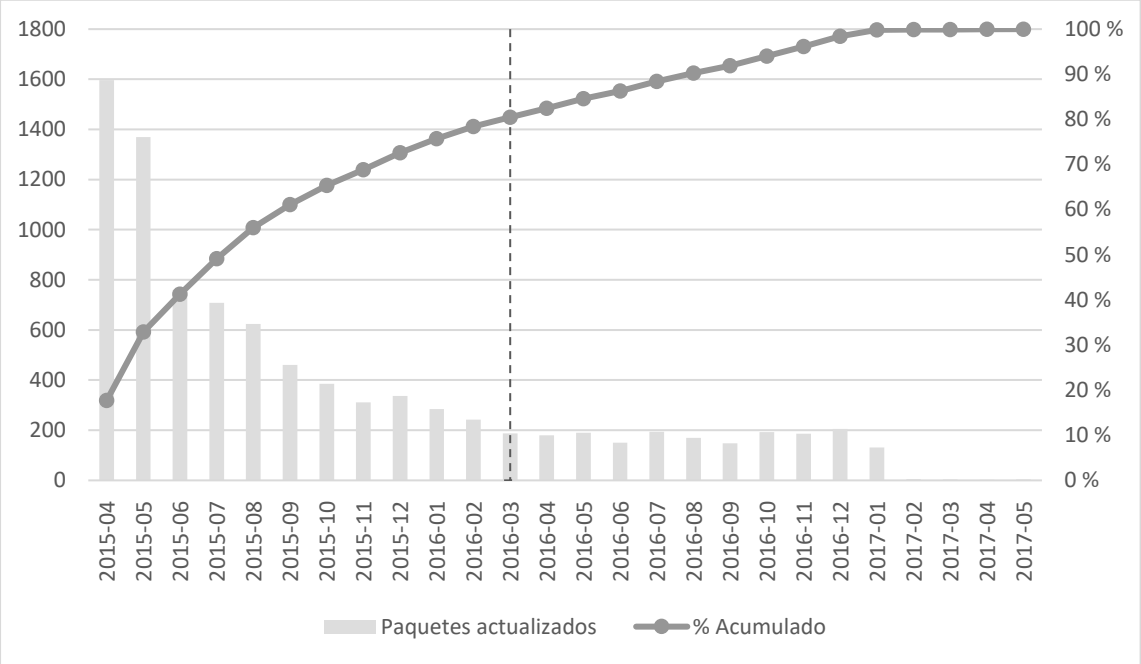
Figura 15. **Paquetes nuevos 2013 - 2015**



Fuente: elaboración propia, empleando Microsoft Word.

La siguiente figura muestra como el 80 % de primeras actualizaciones mayores se completó durante los doce meses comprendidos entre el 2015 y 2016. El 20 % restante se completó en los diez meses siguientes comprendidos entre 2016 y 2017.

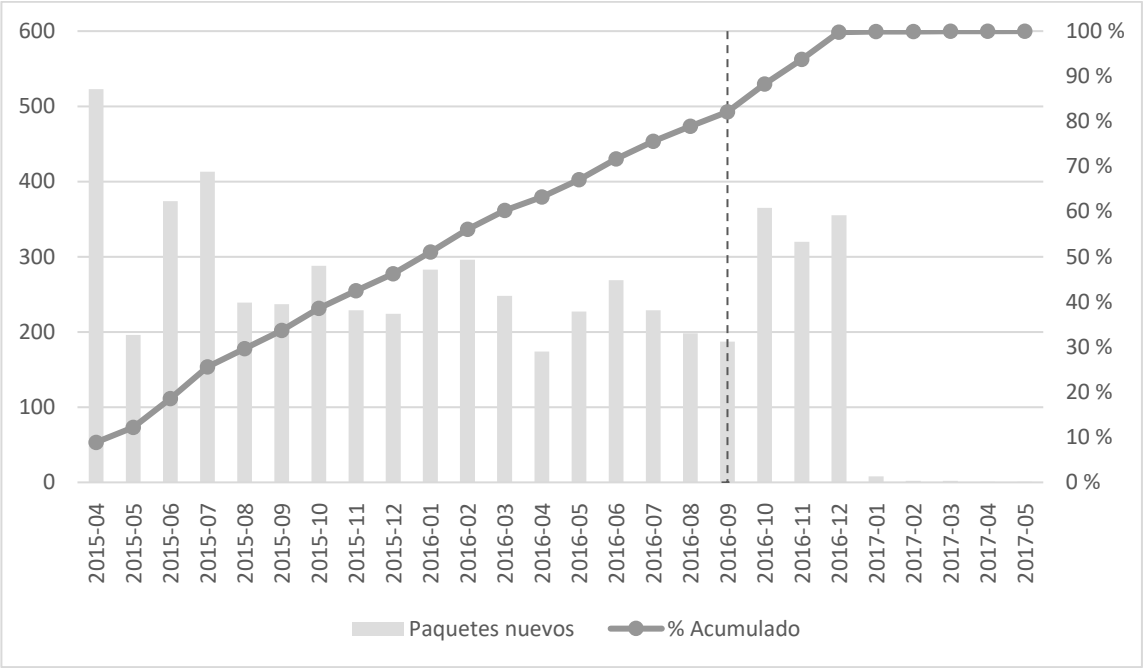
Figura 16. **Primera actualización mayor 2015 - 2017**



Fuente: elaboración propia, empleando Microsoft Word.

El ritmo de creación de paquetes nuevos en este ciclo es uniforme, lo que se evidencia en la siguiente gráfica que incluye el porcentaje acumulado mensualmente. A diferencia de ciclos anteriores el 80 % de los paquetes nuevos fueron completados hasta el mes número dieciocho, lo que indica que la producción se mantuvo por más tiempo que en el resto de los ciclos de desarrollo.

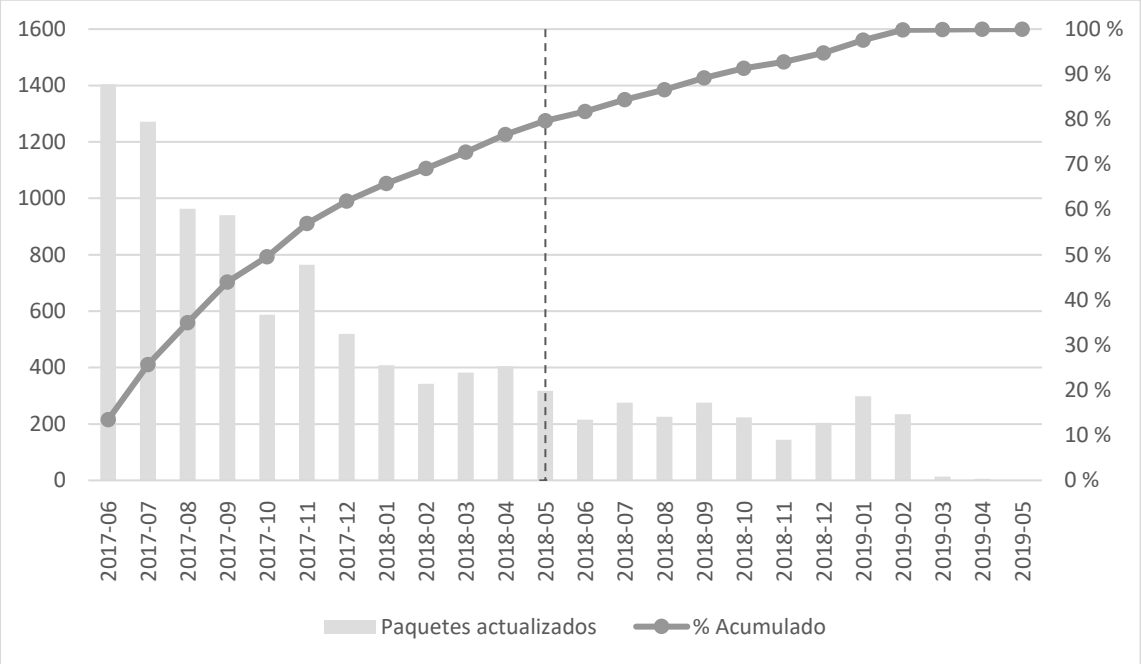
Figura 17. **Paquetes nuevos 2015 - 2017**



Fuente: elaboración propia, empleando Microsoft Word.

El último ciclo de desarrollo analizado en la presente investigación, completo el 80 % de las primeras actualizaciones mayores en doce meses. Como se muestra en la siguiente figura, los primeros siete meses mostraron un mayor nivel de productividad al compararlos con el resto del ciclo de desarrollo que duró en total veinticinco meses:

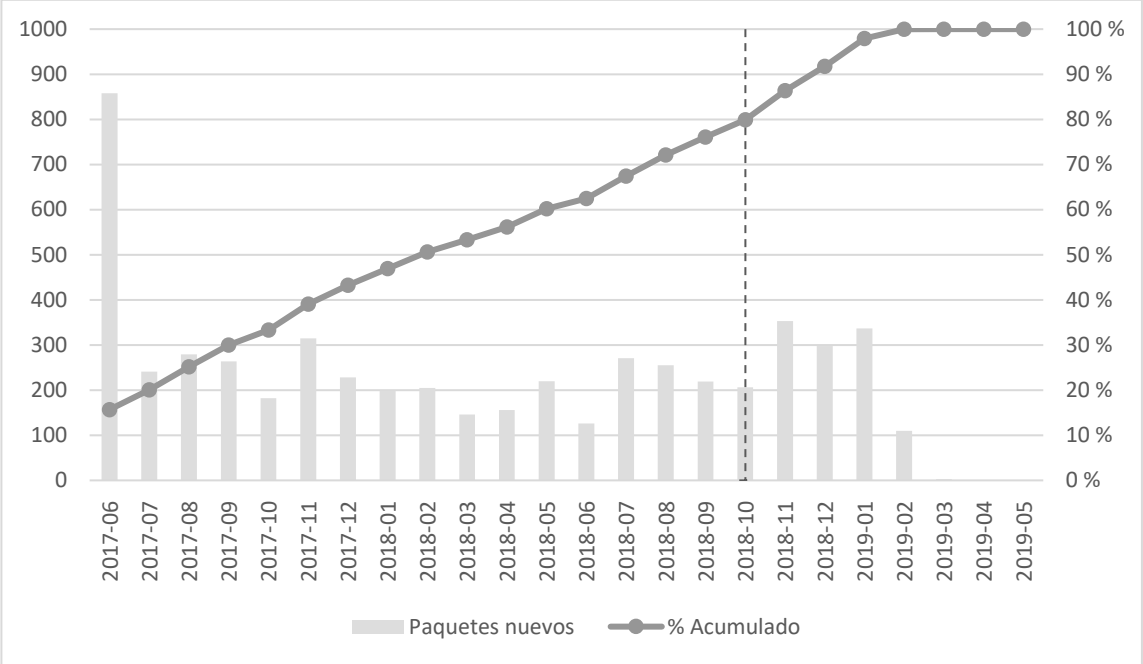
Figura 18. **Primera actualización mayor 2017 - 2019**



Fuente: elaboración propia, empleando Microsoft Word.

La gráfica del porcentaje de paquetes nuevos acumulados mensualmente, que se presenta a continuación, tiene un crecimiento constante que alcanzó el 80 % de los paquetes nuevos al finalizar el mes número diecisiete de los veinticinco que duró el ciclo de desarrollo de 2017 a 2019:

Figura 19. **Paquetes nuevos 2017 - 2019**



Fuente: elaboración propia, empleando Microsoft Word.

Los datos presentados anteriormente hacen ver que el proyecto Debian GNU/Linux tiene la posibilidad de reducir el tiempo invertido en cada ciclo de trabajo gracias a que el equipo de desarrollo logra consistentemente aumentar la producción de actualizaciones mayores al inicio de cada ciclo y al mismo tiempo mantiene una producción constante de paquetes nuevos cada mes. Si se toma como referencia el punto en el que el equipo de desarrolladores ha completado el 80 % del trabajo, se puede observar que el proyecto podría reducir varios meses de trabajo en cada ciclo de desarrollo.

2.4. Errores detectados entre entregas

El proyecto Debian GNU/Linux recibe reportes de errores por parte de sus usuarios y del equipo de desarrolladores. Cada reporte se clasifica de acuerdo con la siguiente escala en orden ascendente[40]:

- Solicitud
- Menor
- Normal
- Importante
- Serio
- Grave
- Crítico

Para formar parte de una entrega final de Debian GNU/Linux un paquete de software debe resolver previamente cualquier reporte de error de nivel serio o superior.

En la siguiente figura se presentan los seis ciclos de desarrollo mostrando la cantidad de reportes de error con un nivel serio o superior, registrados mensualmente, con el objetivo de identificar si existe una tendencia en común.

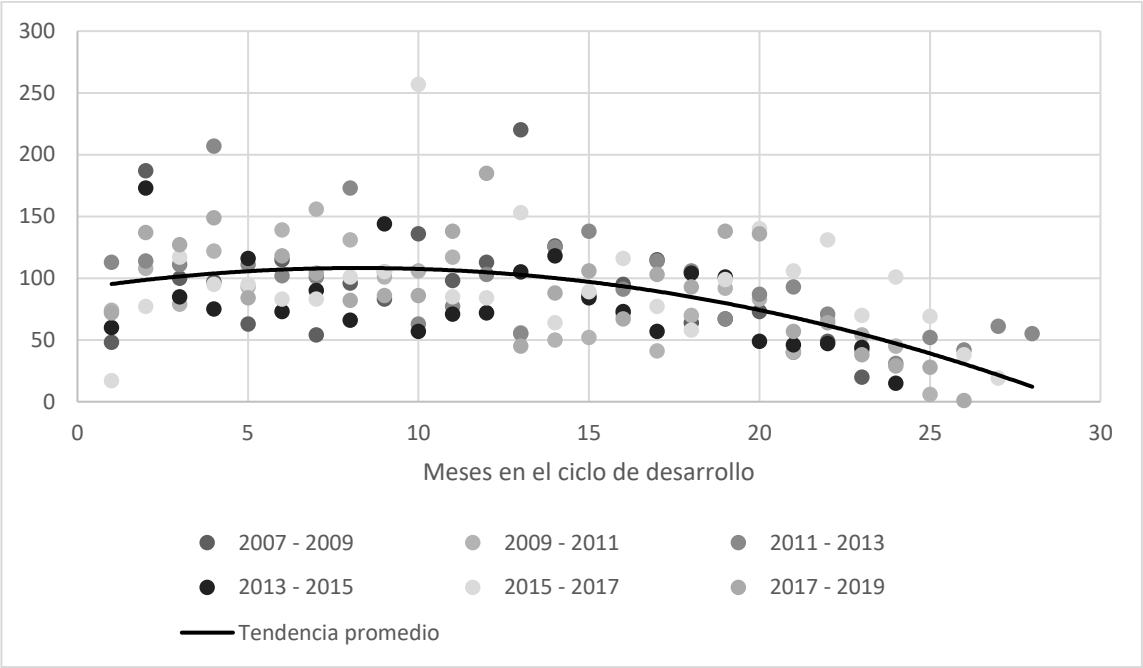
La figura contiene los siguientes elementos:

Puntos: representan la cantidad de errores reportados en un mes.

Línea de tendencia: muestra la tendencia promedio al sobreponer todos los ciclos de trabajo desde el año 2007 al 2019.

[40] JACKSON, Ian; BENHAM, Darren. *Debian BTS - developer info*. https://www.debian.org/Bugs/Developer#severities.

Figura 20. **Reportes de error mensuales**



Fuente: elaboración propia, empleando Microsoft Word.

La tendencia que se presenta en la figura anterior muestra la cantidad de errores reportados mensualmente. Se puede observar que, durante la mayor parte del ciclo de desarrollo, la cantidad reportada de errores es similar pero disminuye cuando se acerca el final del ciclo debido a que el equipo de desarrollo del proyecto Debian GNU/Linux pasa a enfocarse principalmente en la resolución de los errores reportados.

2.5. Análisis de requerimientos de entregas finales

Desde el punto de vista de un integrante individual del equipo de desarrollo de Debian GNU/Linux cada nueva versión de un paquete de software que es completado debe estar 100 % lista para ser entregada a un usuario final.

Sin embargo, las entregas finales a los usuarios se realizan en conjunto, por lo que la entrega de todos los paquetes de software que constituyen una nueva versión de Debian GNU/Linux requiere el completar los siguientes requerimientos[41]:

- Ciclo de tiempo
- Metas de la entrega
- Errores críticos para la entrega

2.5.1. Ciclo de tiempo

El ciclo de tiempo es el requerimiento que se maneja con mayor flexibilidad de los tres requerimientos listados anteriormente y fue establecido oficialmente en el 2009. Consiste en que se tiene planeado el realizar una nueva entrega final de Debian GNU/Linux cada dos años y como se presentó en la sección 2.3 de esta investigación, se ha mantenido un promedio de 24 meses entre cada entrega final[42].

2.5.2. Metas de la entrega

Las metas de la entrega consisten en el conjunto de nuevas funcionalidades que el equipo de desarrollo de Debian GNU/Linux desea implementar como parte de la siguiente entrega final. Luego de que una nueva meta para la siguiente entrega es aceptada, se procede a crear los reportes de error correspondientes en cada paquete de software que requiera cambios para lograr el objetivo. Dichos

[41] KRAFFT, Martin F. *The Debian system: concepts and techniques*. p. 43.
[42] REICHLE, Meike. *Debian to adopt time-based releases*. https://lwn.net/Articles/344007/.

reportes de error constituyen la herramienta para dar seguimiento a las metas de la entrega.

2.5.3. Errores críticos para la entrega

Los errores críticos para la entrega son el requerimiento más importante para la entrega. Todo reporte de error de nivel serio, grave o crítico es considerado un requisito para la siguiente entrega. Los últimos meses del ciclo de desarrollo son utilizados de forma exclusiva para que el equipo de desarrollo de Debian GNU/Linux se dedique a la resolución de estos reportes de error. Aun así, hay cierto grado de flexibilidad que permite postergar o cambiar el nivel de un error crítico, con el fin de completar la entrega final de una nueva versión de Debian GNU/Linux.

2.6. Factores de productividad identificados

El análisis de las métricas realizado en las secciones anteriores permite identificar la productividad del equipo de desarrollo de Debian GNU/Linux no es aprovechada al máximo. Se observaron principalmente dos factores que tienen un impacto negativo en la productividad del proyecto Debian GNU/Linux:

- Tiempos de espera arbitrarios
- Proceso de entrega simultáneo

2.6.1. Tiempos de espera arbitrarios

Debido a que el tiempo que se espera antes de iniciar el proceso para realizar una nueva entrega es arbitrario, esto resulta, como se observó anteriormente, causando que entre un 30 % y un 38 % de los paquetes reciban por lo menos una actualización mayor que no es aprovechada por el usuario final.

Adicionalmente, al comparar la productividad mensual de actualizaciones mayores y creación de paquetes nuevos se observa que existe la posibilidad de reducir el tiempo del ciclo de desarrollo, para aprovechar de mejor manera la capacidad productiva del equipo de desarrollo de Debian GNU/Linux.

2.6.2. Proceso de entrega simultáneo

El proceso de entrega es simultáneo, es decir, que todos los paquetes de software se entregan al mismo tiempo. En el caso específico de Debian GNU/Linux, esto es una desventaja debido a que cada paquete de software tiene un ritmo de actualización diferente, ya que cada mes son completados múltiples paquetes nuevos, sin embargo, éstos no son entregados hasta que se coordina una entrega simultánea de todos los paquetes de software. La productividad del equipo de desarrollo de software de Debian GNU/Linux puede ser mejor aprovechada si se busca un esquema de entregas alternativo que tome en cuenta estos ritmos de producción de actualizaciones y paquetes nuevos.

En las siguientes secciones de esta investigación se evaluarán otras alternativas para la gestión de entregas de software y se evaluará la posibilidad de que tengan una incidencia positiva sobre la productividad del proyecto Debian GNU/Linux.

3. MEJORES PRÁCTICAS EN LA GESTIÓN DE ENTREGAS DE SOFTWARE

Los equipos de desarrollo de software pueden pasar meses trabajando en un nuevo proyecto, sin embargo, el software solo puede cosechar su valor cuando alcanza las manos de las personas para las que ha sido creado. La gestión de entregas de software pretende facilitar este proceso[43] y consiste en la planeación, administración y control mediante el cual, nuevas versiones de aplicaciones o sistemas operativos son entregadas al usuario final[44].

Con el paso del tiempo, los equipos de desarrollo de software han adoptado un conjunto de mejores prácticas y esquemas estructurados para la gestión de entregas de software, que intentan adaptarse a las necesidades específicas de cada proyecto y equipo de desarrollo.

En las siguientes secciones se detallan las mejores prácticas y los esquemas más utilizados para la gestión de entregas de software.

3.1. Gestión de entregas de los estándares ITIL e ISO 20000

La presente sección aborda al mismo tiempo ITIL e ISO 20000 debido a sus similitudes. Como se detalla más adelante, la norma ISO 20000 está basada en ITIL y aunque la norma ISO 20000 agrega un camino para la certificación de las

[43] ARORA, Tarun. *Microsoft team foundation server 2015 cookbook*. p. 218.
[44] BALGROSKY, Jean A. *Essentials of health information systems and technology*. p. 79.

organizaciones que la implementan, para los propósitos de esta investigación, ambas pueden ser tratadas en conjunto.

3.1.1. ITIL

> La biblioteca de infraestructura de tecnología de la información *o Information Technology Infrastructure Library* es también conocida como ITIL, por sus siglas en inglés y consiste en un conjunto de mejores prácticas para la gestión de servicios de tecnologías de la información que son publicadas con el objetivo de proveer una guía para todo tipo de organizaciones sin importar sus objetivos, estrategias o tipo de trabajo[45].

"Desde la primera versión de ITIL, que fue publicada en la década de 1980 por el gobierno del Reino Unido, se hace énfasis en que ITIL debía ser implementado adaptándolo a las necesidades específicas de cada organización"[46].

3.1.2. ISO 20000

> También conocida como ISO/IEC 20000 fue desarrollada en el año 2005, basándose en ITIL, debido al reconocimiento recibido alrededor del mundo para ITIL. La norma ISO 20000 surgió con el objetivo de proveer a las organizaciones que administran servicios de tecnologías de la información una especificación medible, auditable y que incluyera las mejores prácticas presentes en ITIL[47].

"Vale la pena remarcar que la norma ISO 20000 requiere el uso del ciclo de Deming, que consiste en aplicar cuatro pasos de forma iterativa"[48]:

- Planificar: establecer, documentar y acordar los objetivos y procesos
- Hacer: implementar según lo planeado
- Verificar: determinar si las actividades obtuvieron el resultado esperado

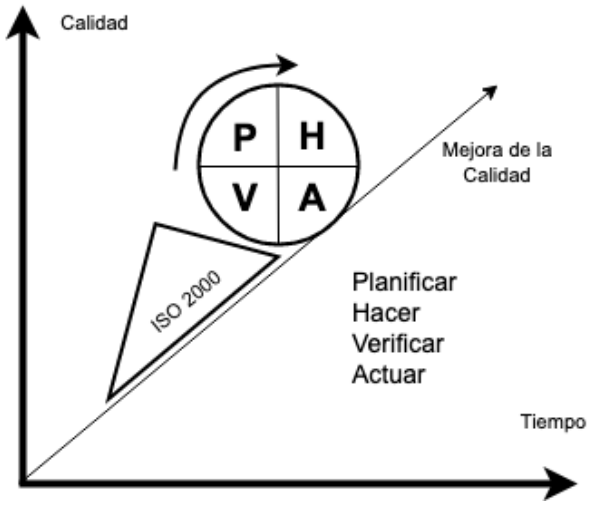
---

[45] THE STATIONERY OFFICE. *ITIL Foundation*. p. 8.
[46] HUNNEBECK, Lou. *Key element guide ITIL service design*. p. 1-3.
[47] LAPORTE, Claude Y. y APRIL, Alain. *Software quality assurance*. p. 141.
[48] BON, Jan van. *ISO/IEC 20000 An Introduction*. p. 5.

- Actuar: mejorar continuamente el desempeño y resolver cualquier problema encontrado en la verificación

La siguiente figura ilustra el uso del ciclo Deming como una herramienta para que las organizaciones mantengan la búsqueda de aumentar su calidad de forma continua:

Figura 21. **Ciclo Deming**



Fuente: BON, Jan. ISO/IEC 20000 *An Introduction*. p. 6.

3.1.3. Especificaciones de la gestión de entregas

Una entrega, según la definición de ITIL, puede estar formada por diferentes componentes de software o infraestructura que, trabajando juntos, proveen una nueva funcionalidad, o bien, un cambio a una funcionalidad actual.

Los componentes que forman parte de una entrega pueden ser desarrollados internamente o proveídos por un tercero para ser integrados como parte de un sistema existente[49].

[49] THE STATIONERY OFFICE. *ITIL Foundation*. p. 9-11.

ITIL define las siguientes mejores prácticas para la gestión de entregas de software:

- Planificar el tamaño, alcance y contenido de cada entrega.
- Diseñar la frecuencia y el contenido para satisfacer las necesidades y expectativas de los clientes y los usuarios.
- Asegurarse que las entregas sean puestas a disposición de los usuarios en una forma controlada.
- Obtener los componentes desarrollados para la próxima entrega siguiendo un proceso previamente definido.
- Tener en cuenta la documentación de los cambios incluidos, errores esperados, guías de usuarios y guías de soporte.

> La norma ISO 20000 remarca la importancia de la planificación y la coordinación para lograr gestionar entregas de software de forma exitosa, reduciendo los riesgos para el cliente. En su sección 8.5.3 establece los siguientes criterios para verificar que la gestión de entregas se está realizando de acuerdo a la norma
>
> - Documentar la frecuencia y los tipos de entrega.
> - Planificar con antelación cada entrega con el acuerdo de todas las partes implicadas.
> - Tener en cuenta la posibilidad de revertir o cancelar la entrega al cliente como parte del proceso de entrega[50].

Como se puede observar en los párrafos anteriores, el marco de trabajo ITIL plantea un conjunto de mejores prácticas que son compatibles con los requisitos de la norma ISO 20000, tomando en cuenta esto, el autor Michael Kunas[51] propone que una empresa que tenga como objetivo gestionar entregas de software de acuerdo con la norma ISO 20000, planee por adelantado todos los aspectos de la gestión de entregas, incluyendo:

[50] International Organization for Standardization. *ISO/IEC 20000-1:2018*. https://www.iso.org/standard/70636.html.

[51] KUNAS, Michael. *Implementing Service Quality based on ISO/IEC 20000: A Management Guide.* p. 91.

- Política de entregas
- Plan de ejecución
- Diseño, construcción y configuración
- Verificación y aceptación
- Documentación

3.1.3.1. Política de entregas

La política de entregas es el documento en donde se establecen de forma general los tipos de entregas y la frecuencia con que se realizarán, además de esto debe incluir los siguientes elementos

- La forma específica en que se agruparán los cambios que forman parte de una entrega.
- Roles y responsabilidades en la gestión de la entrega, incluyendo el listado de personas con la autoridad para decidir si una entrega cumple con los requisitos esperados.
- Descripción de los procesos que serán automatizados
- Especificación del proceso de verificación y aceptación de la entrega por parte del cliente[52].

3.1.3.2. Plan de ejecución

El plan de ejecución tiene como objetivo asegurar que los cambios a los sistemas afectados sean acordados, autorizados y planeados antes de ser ejecutados. El plan debe contener los siguientes elementos:

- Fecha y descripción del proceso
- Listado de cambios y errores conocidos
- Procesos relacionados
- Proceso para revertir los cambios
- Proceso de aceptación

[52] KUNAS, Michael. *Implementing Service Quality based on ISO/IEC 20000: A Management Guide.* p. 91-102.

- Medios de comunicación a utilizar durante la entrega
- Dependencias y riesgos

3.1.3.3. Diseño, construcción y configuración

La entrega de un nuevo componente de software o conjunto de cambios debe ser el resultado de un proceso diseñado previamente, de manera que el producto cumpla con los estándares requeridos por el cliente, asegure su integridad durante cada una de las fases, se identifiquen los riesgos y se tomen medidas para reducir dichos riesgos. Para lograr incrementar la efectividad y eficiencia la construcción y configuración debe ser automatizada tanto como sea posible.

3.1.3.4. Verificación y aceptación

Cada entrega debe recibir el visto bueno del personal autorizado y entre los aspectos que se deben verificar previo a la aceptación se tienen los siguientes:

- Asegurarse que la entrega ha sido probada y dichas pruebas fueron llevadas a cabo de forma satisfactoria de acuerdo con el cliente y con el equipo de desarrollo.
- Asegurarse que la entrega está compuesta exclusivamente por los elementos de software y hardware esperados.
- Verificar que el ambiente de pruebas corresponde al ambiente de producción.
- Verificar que la entrega ha sido completada luego de que el proceso se ha concluido.

3.1.3.5. Documentación

Cada entrega deberá estar acompañada de una documentación adecuada que incluya todos los detalles correspondientes, tales como:

- Visión general del sistema
- Proceso de instalación
- Planes de emergencia
- Planes de capacitación
- Listado de cambios
- Problemas esperados

3.2. Estándar ANSI/PMI 99-001-2017 aplicado a la gestión de entregas de software

> El estándar ANSI/PMI 99-001-2017 contiene una serie de normas y mejores prácticas para la gestión de proyectos. El estándar define un proyecto como un esfuerzo que se realiza por un tiempo definido, con el objetivo de crear un producto, servicio o resultado único y especifica que la gestión de proyectos es la aplicación de conocimientos, habilidades y herramientas para cumplir con los objetivos de un proyecto[53].

De acuerdo con el estándar ANSI/PMI 99-001-2017, todas las actividades que deben realizarse para completar un proyecto pueden ser catalogadas en dos ejes principales: áreas de conocimiento y grupos de procesos.

[53] Project Management Institute. *A guide to the project management body of knowledge*. p. 4-12.

3.2.1. Áreas de conocimiento

Las áreas de conocimiento de la gestión de proyectos son campos o áreas de especialización que se emplean comúnmente en la gestión de proyectos. Un área de conocimiento es un conjunto de procesos asociados con un tema específico que forma parte de la gestión de proyectos. El estándar ANSI/PMI 99-001-2017 enumera las siguientes áreas de conocimiento:

- Integración: esta área incluye los procesos para identificar, definir, combinar, unificar y coordinar las distintas actividades del proyecto y asignar recursos y balancear los objetivos del mismo.

- Alcance: se encarga de asegurarse que el proyecto incluye todo el trabajo que es requerido, pero solamente el trabajo requerido para alcanzar los objetivos del proyecto.

- Tiempo: incluye los procesos requeridos para lograr que el proyecto sea completado a tiempo.

- Costo: esta área de conocimiento incluye la planificación, estimación, presupuesto, financiamiento y control de costos de manera que el proyecto sea completado sin salirse del presupuesto aprobado para el mismo.

- Calidad: se encarga de incorporar las políticas de calidad propias de la organización, para lograr que los productos finales cumplan con los requisitos de calidad esperados.

- Recursos: incluye los procesos que se encargan de identificar, adquirir y gestionar todos los recursos que sean necesarios para completar el proyecto de forma exitosa.

- Comunicaciones: se encarga de la planificación, recopilación, distribución, almacenamiento y monitoreo de toda la información del proyecto, asegurando que todos los involucrados en el proyecto obtengan la información que necesitan de forma oportuna y adecuada.

- Riesgo: incluye la identificación, análisis, planificación e implementación de posibles respuestas y el monitoreo del riesgo en un proyecto, con el objetivo de optimizar las posibilidades de éxito del proyecto.

- Adquisiciones: incluye los procesos necesarios para adquirir productos, servicios o resultados requeridos para llevar a cabo el proyecto.

- Partes interesadas: esta área incluye los procesos necesarios para identificar a las personas, grupos u organizaciones que podrían tener un impacto o bien ser impactadas por el proyecto. También incluye el análisis de las expectativas de las partes interesadas y el desarrollo de estrategias de gestión efectivas para involucrarlas en la toma de decisiones y la ejecución del proyecto.

3.2.2. Grupos de procesos

El estándar describe cinco grupos que engloban los distintos procesos que son empleados para alcanzar los objetivos del proyecto:

- Procesos de iniciación: son los procesos que se realizan para definir un nuevo proyecto o una nueva fase de un proyecto existente, incluyendo su respectiva autorización.

- Procesos de planificación: incluye a los procesos que se encargan de establecer el alcance del proyecto, definir el curso de acción y refinar los objetivos del proyecto.
- Procesos de ejecución: son los procesos que se llevan a cabo para completar el trabajo definido en el plan del proyecto.

- Procesos de monitoreo y control: incluye a los procesos que son necesarios para dar seguimiento, evaluar y regular el progreso del proyecto. Estos procesos también identifican si se requieren cambios en el plan.

3.2.3. Gestión de entregas de software

Al aplicar los principios y mejores prácticas del estándar ANSI/PMI 99-001-2017 Aguilh[54] define los siguientes procesos para la gestión de entregas de software:

- Empaquetado de la entrega: es el proceso mediante el cual se negocia y finalmente se decide lo que será incluido en la entrega de software.

- Documentación: este proceso establece el mecanismo mediante el cual todos los documentos acerca de los cambios incluidos en una entrega son

[54] AGUILH, Franck. *Understanding software release management*. *Proceedings of the Project Management Institute Annual Seminars & Symposium*. p. 52.

distribuidos a individuos u organizaciones que necesitan tener acceso a los mismos.

- Control del cambio: es el proceso que se sigue para la notificación autorización, documentación e implementación de cualquier cambio al paquete de funcionalidades que se habían aprobado con anterioridad para la siguiente entrega de software.

Pruebas: este proceso incluye la coordinación de la participación del cliente para que este pueda probar las nuevas funcionalidades del sistema previo a la entrega. También incluye la definición de los lineamientos mediante los cuales las pruebas se llevarán a cabo.

Despliegue: este proceso es el conjunto de procedimientos mediante el cual se realiza la entrega del software al cliente, incluyendo la coordinación y comunicación con todas las partes interesadas y las pruebas finales posteriores al despliegue.

A continuación, se presenta la catalogación de cada uno de los procesos con respecto a su grupo y las áreas de conocimiento involucradas del estándar ANSI/PMI 99-001-2017:

Tabla XIV. **Procesos para la gestión de entregas de software**

Proceso	Grupo	Áreas de conocimiento
Empaquetado	Iniciación, planificación	Integración, comunicaciones, alcance, costo, recursos
Documentación	Monitoreo y control	Comunicaciones, riesgo, recursos, partes interesadas
Control del cambio	Monitoreo y control	Integración
Pruebas	Ejecución	Calidad
Despliegue	Ejecución, cierre	Tiempo, riesgo

Fuente: AGUILH, Franck. *Understanding software release management.* p. 52.

El seguir estos procesos aplicando los principios del estándar ANSI/PMI 99-001-2017 permite que la gestión de la entrega de software sea completada a tiempo, sin impactar la funcionalidad existente, satisfaciendo los requerimientos del cliente y sin exceder el presupuesto acordado.

3.3. Gestión de la aceptación de cambios y la transición en el marco de trabajo COBIT 2019

"COBIT es un marco de trabajo que provee un conjunto de mejores prácticas y procesos para la gobernanza de tecnologías de la información"[55]. Los cinco principios fundamentales que rigen el marco de trabajo COBIT son:

- Separar la gobernanza de la gestión.
- Cumplir con las necesidades de las partes interesadas.
- Cubrir a la compañía de extremo a extremo.
- Aplicar un marco de trabajo integrado y único.
- Permitir un enfoque holístico.

> En cuanto a la gestión de entregas de software COBIT define el proceso llamado: gestión de la aceptación de cambios y la transición, dicho proceso conlleva las siguientes actividades
>
> Plan de implementación
> Plan de migración de procesos de negocio
> Pruebas de aceptación
> Entorno de pruebas
> Ejecución de pruebas
> Paso a producción
> Soporte
> Evaluación[56].

3.3.1. Plan de implementación

> Se debe crear un plan que incluya toda la secuencia de acciones de implantación, los criterios que se utilizarán para su aceptación y las dependencias, dicho plan debe ser aprobado por todas las partes interesadas incluyendo el ámbito de negocio y técnico. El plan debe incluir la secuencia de acciones para revertir la implantación y asegurar que se han considerado los riesgos técnicos y de negocio[57].

[55] COBIT. *Control Objectives for Information Technologies ISACA*. https://www.isaca.org/resources/cobit.

[56] ISACA. *COBIT 2019 Framework Governance and Management Objectives*. p. 151-228.

[57] Ibíd.

3.3.2. Plan de migración de procesos de negocio

Previo a la implantación del nuevo sistema o cambios en el sistema se deben considerar todos los procesos de negocio que serán afectados, respaldando todos los sistemas y datos, al mismo tiempo que se mantienen registros de auditoría de manera que los resultados puedan ser auditados.

También se debe coordinar el tiempo en que el proceso será completado, con el objetivo de que se dé una transición continua sin pérdidas para el negocio o de ser necesario, coordinar un paro de las operaciones del negocio.

3.3.3. Pruebas de aceptación

Esta actividad tiene como resultado un plan de pruebas que debe cubrir los aspectos funcionales y los riesgos del proyecto, así como identificar las fases de prueba adecuadas dependiendo de las operaciones del negocio. El plan también debe identificar todos los recursos que se necesitan para la ejecución de las pruebas y la evaluación de los resultados, comunicándolo a todas las partes interesadas.

3.3.4. Entorno de pruebas

Como resultado de esta actividad se debe establecer un entorno de pruebas que tome en cuenta las características de las operaciones del negocio en cuanto a carga de trabajo, seguridad, privacidad de la información, infraestructura y rendimiento esperado.

3.3.5. Ejecución de pruebas

Previo a la implantación de la nueva entrega de software se deben ejecutar todas las fases de pruebas de acuerdo con el plan de pruebas establecido, llevando un registro de los resultados. Las pruebas deben ser diseñadas y ejecutadas por un grupo de personas independiente del grupo que desarrolló el software, involucrando a los encargados de procesos de negocio y usuarios finales. Es necesario que también se pongan a prueba los planes de contingencia o procesos para revertir la implantación de los cambios. Finalmente se debe obtener la aprobación de todas las partes interesadas luego de la entrega de los resultados.

3.3.6. Paso a producción

Esta actividad consiste en la ejecución de todos los pasos definidos en el plan de implementación, teniendo como resultado final la entrega del software al negocio y su equipo de operaciones. De ser apropiado, la nueva versión del software se puede ejecutar en paralelo con la versión anterior, para comparar sus resultados. En caso de tener errores significativos se debe revertir el trabajo realizado de acuerdo al plan.

3.3.7. Soporte

Se deben soportar las operaciones del negocio por un período acordado previamente y de ser necesario proporcionar recursos adicionales, todo esto con el objetivo de abordar cualquier incidencia y asegurarse que la nueva solución entregada al cliente está funcionando de forma estable.

3.3.8. Evaluación

Luego de completar la implantación del sistema se debe hacer una revisión para evaluar los resultados tomando en cuenta a los encargados de procesos del negocio, el rendimiento actual del sistema nuevo o modificado en comparación al rendimiento previsto. Se deben también incluir los siguientes elementos:

- Estabilidad del sistema.
- Impactos inesperados.
- Riesgos que fueron mitigados.
- Eficacia y eficiencia.
- Cumplimiento de requisitos corporativos.
- Beneficios obtenidos.
- Plan de acción para resolver problemas identificados.

3.4. Mejores prácticas DevOps para la gestión de entregas de software

El termino DevOps proviene de combinar las palabras en inglés *development* que significa desarrollo y *operations* que significa operaciones[58] y hace referencia a un conjunto de prácticas en las que se propone que los equipos de negocio, desarrollo de software y operaciones colaboren de manera continua para asegurarse que las soluciones de tecnologías de la información estén disponibles a tiempo, cumplan con las expectativas y funcionen sin interrupciones[59].

[58] MICROSOFT. *¿Qué es DevOps?* https://azure.microsoft.com/es-mx/overview/what-is-devops/.

[59] SAMER I, Mohamed. *Software release management evolution-comparative analysis across agile and DevOps continuous delivery*. *International Journal of Advanced Engineering Research and Science*. p. 52-59.

El enfoque adoptado por las prácticas que forman parte de DevOps sigue estos tres principios[60]:

- Crear equipos que se dirigen a sí mismos en lugar de tener una jerarquía de mando centralizada en una sola persona.
- Preferir múltiples iteraciones que permitan corregir el rumbo en lugar de pretender hacer una implementación perfecta en la primera entrega.
- Automatizar los procesos.

La gestión de entregas al seguir los tres principios mencionados anteriormente debe integrarse con todo el proceso de desarrollo de software, desde el momento en que se decide implementar un cambio o una nueva funcionalidad, sus pruebas, aceptación y el despliegue de la solución, por lo anterior, al implementar una gestión de entregas de software se deben considerar las siguientes prácticas[61]:

Implementar un ciclo de desarrollo de productos que tenga una gestión integrada de todas las fases que el negocio considere necesarias, desde su concepción hasta su entrega.

Dar seguimiento al estado de cada despliegue de cambios en cada uno de los ambientes.

Auditar el historial de todas las actividades de cada unidad de trabajo que forma parte de una entrega.

Automatizar cada una de las etapas incluyendo la entrega final.

Empoderar a los equipos para que puedan realizar entregas automatizadas de forma repetible, confiable y manteniendo un alto nivel de seguimiento a través de métricas y evaluación de resultados.

[60] VADAPALLI, Sricharan. *DevOps: Continuous Delivery, Integration, and Deployment with DevOps*. p. 11-28.

[61] Ibíd.

Proveer acceso para que las personas autorizadas puedan aprobar cada uno de los cambios de forma eficaz y eficiente.

Finalmente es importante remarcar que la implementación de las prácticas mencionadas anteriormente requiere un alto grado de colaboración entre los distintos equipos de la organización. Para ello se [62] propone que es indispensable el cuantificar el valor de cada cambio que se realiza, definiéndolo de una forma clara que esté en sintonía con los objetivos de la organización.

Por ejemplo, hacer del conocimiento de todos los equipos a cuánto asciende el valor monetario de inactividad por minuto de los sistemas de tecnologías de la información de la empresa. Este valor monetario provee un lenguaje común a todos los equipos, para expresar el impacto que tiene la entrega de una nueva versión del software que reduzca la cantidad de minutos de inactividad reportada mensualmente.

3.5. Gestión de entregas continuas

La gestión de entregas continuas es un conjunto de buenas prácticas que tiene como objetivo el proveer toda la infraestructura para que un equipo de desarrollo de software pueda producir nuevas funcionalidades o cambios al software existente en ciclos cortos, asegurando que el software puede ser entregado al usuario final en cualquier momento.

> El concepto principal en la implementación de entregas continuas es el concepto llamado: tubería de entregas, el cual establece cinco etapas por las que todo cambio, que es introducido al sistema, debe pasar en un tiempo relativamente corto y de forma repetitiva. Las etapas definidas son[63]:

[62] FAYER, Leon. *DevOps and Business*. p. 1-3.

[63] FARLEY, Dave; HUMBLE, Jez. *Continuous delivery: reliable software releases through build, test, and deployment automation.* p. 103-142.

Etapa de consolidación: cada vez que un integrante del equipo de desarrollo completa un cambio, esto debe iniciar de forma automatizada los procesos de construcción del software, pruebas unitarias, análisis del código y generación de artefactos.

Pruebas de aceptación automatizadas: luego de que el software ha completado exitosamente la etapa anterior, debe ser probado de forma automática en un ambiente similar al ambiente en el que será utilizado, esto con el objetivo de verificar que cumple con la funcionalidad requerida por el cliente.

Pruebas automáticas de desempeño: en la siguiente etapa el software es sometido a pruebas automáticas de carga, que puedan verificar que el software es capaz de soportar la carga de trabajo requerida en el ambiente de producción.

Pruebas manuales: en esta fase, luego de que el software es desplegado de forma automática a un ambiente de pruebas, se solicita la intervención de una persona para realizar pruebas manuales de uso del sistema, explorando múltiples casos de uso o bien tomando esta oportunidad para hacer una demostración del sistema a los clientes. Cuando la persona está satisfecha, podrá ingresar al sistema los resultados para que el software pueda pasar a la siguiente fase.

Entrega: en esta fase el software está disponible para ser entregado al usuario final de forma automática, en cuanto los integrantes del equipo así lo decidan. Esta etapa también conlleva el monitoreo del funcionamiento del software y la capacidad de revertir la entrega de ser necesario.

Como se observa en cada una de las etapas que se mencionaron, la gestión de entregas continuas pone un énfasis en la automatización de los procesos y en el proveer herramientas para hacer eficientes las pruebas y aprobaciones manuales. Al implementar la gestión de entregas continuas se busca proveer a las organizaciones la posibilidad de aprovechar cada cambio realizado por el equipo de desarrollo de forma óptima y tan rápida como el negocio lo requiera, gestionando el riesgo de forma adecuada.

3.6. Estrategias de gestión de entregas en proyectos de código abierto

La gestión de entregas en proyectos de código abierto suele seguir muchas de las etapas especificadas en las secciones anteriores, pero además suele seguir un criterio específico en torno al cual, cada nueva versión del proyecto, es entregada

a los usuarios finales, principalmente se pueden identificar los siguientes enfoques[64]:

3.6.1. Tiempo

Al utilizar este criterio se define un período fijo, tras el cual todas las funcionalidades que han sido completadas de forma satisfactoria son incluidas en una nueva versión del software y son entregadas al usuario final. Es importante remarcar que en proyectos complejos es común observar diferentes tiempos de entrega para diferentes componentes del software[65].

3.6.2. Funcionalidad

Al inicio del ciclo de desarrollo se establecen una serie de funcionalidades como el principal objetivo de la nueva versión del software. Se considera que la nueva versión esta lista para ser entregada a los usuarios solamente, cuando los objetivos se han completado, sin importar la cantidad de tiempo que esto tome[66].

3.6.3. Entregas continuas

En este tipo de entregas, el equipo de desarrollo pone a disposición de sus usuarios cada cambio en el software conforme es completado y luego de que cumple con los criterios de aceptación definidos por el proyecto.

3.7. Gestión de entregas por tipo de software

A medida que el software se ha especializado para funcionar en ambientes específicos, se ha tenido como resultado una gestión de entregas de software

[64] TEIXEIRA, José Apolinário; KARSTEN, Helena. *Managing to release early, often and on time in the OpenStack software ecosystem*. *Journal of Internet Services and Applications*. p. 7.

[65] *ce*KHOMH, Foutse, DHALIWAL, Tejinder, ZOU, Ying, et al. *Do faster releases improve software quality? An empirical case study of Mozilla Firefox*. p. 179-188.

[66] MICHLMAYR, Martin. *Quality Improvement in Volunteer Free and Open-Source Software Projects – Exploring the Impact of Release Management*. p. 41.

específica para cada tipo de ambiente. A continuación, se tratan algunos ejemplos de dichas adaptaciones:

3.7.1. ERP

Los sistemas de planificación de recursos empresariales, conocidos por sus siglas inglés ERP que significan: *Enterprise Resource Management*. Los sistemas ERP consisten en una solución de software integrada por un conjunto de módulos que proveen soporte a todas las funciones de una empresa, incluyendo: contabilidad, recursos humanos, materiales, efectivo, producción y órdenes de compra[67].

La gestión de entregas de software para sistemas ERP debe responder a las necesidades del negocio y para alcanzar dicho objetivo se deben implementar entregas periódicas utilizando el concepto de carriles de prioridad, con distintas velocidades y clasificando cada entrega de software según su contenido y la urgencia para el negocio.

A continuación, se presenta un ejemplo de cuatro carriles de prioridad:

Emergencias: cambios críticos para el negocio, que deben ser entregados rápidamente y debido a la urgencia estos pueden ser entregados diariamente.

Entregas menores: son cambios que incluyen resolución de errores y cambios de configuración y tienen un impacto bajo en los procesos del negocio y pueden tener una frecuencia de entrega semanal.

Entregas medianas: este tipo de entregas incluye cambios que tienen un impacto medio en los procesos de negocio, como por ejemplo mejoras al software y cambios solicitados que pueden ser entregados una vez al mes.

Entregas mayores: como su nombre lo indica, este tipo de entregas incluye proyectos mayores incluyendo actualizaciones al mismo software que provee la funcionalidad ERP y puede ser realizada una vez cada tres meses[68].

3.7.2. Aplicaciones móviles

La gestión de entregas de software en aplicaciones móviles tiene un componente particular que influye en el tipo de esquema elegido por los equipos de desarrollo, dicho componente es la tienda de software, o como se le conoce en inglés *app store*, la cual provee un mayor nivel de control en la publicación de nuevas versiones de software a los usuarios finales.

[67] SAMARA, Tarek. *ERP and Information Systems*. p. 1.
[68] KALAIMANI, Jayaraman. *SAP project management pitfalls*. p. 115-136.

La gestión de entregas de software en aplicaciones móviles tiene las siguientes etapas.

Alfa: en esta etapa se prueba la aplicación dando acceso a un conjunto reducido de usuarios dentro de la organización con el objetivo de detectar errores tan pronto como sea posible.

Beta: en esta etapa el proveedor de la tienda de software realiza una verificación de la aplicación, asegurándose que el software cumple con las políticas del proveedor y seguidamente es puesto a disposición de un grupo de usuarios limitado.

Producción: luego de cumplir exitosamente con las dos fases anteriores el software pasa a estar disponible para todos los usuarios.

Gracias a las capacidades de distribución que proveen las tiendas de software, se pueden tomar decisiones basadas en marketing, las aplicaciones móviles pueden utilizar como criterio para elegir el momento de realizar una nueva entrega lo siguiente:

Selección de países: la empresa a cargo del desarrollo suele ejecutar una nueva entrega con cada nuevo país soportado, o bien, con cada nueva funcionalidad creada para el país específico.

Campañas de marketing: las entregas pueden estar delimitadas por las campañas de publicidad, de manera que las nuevas funcionalidades entregadas en la aplicación sean promovidas en medios de comunicación y tengan el mayor impacto posible[69]:

3.7.3. Aplicaciones SaaS

El término SaaS son las siglas en inglés de *software as a service*, que traducido significa: software como un servicio. Este tipo de software sigue un modelo de negocio diferente, en el que los clientes acceden al software a través del internet sin tener que preocuparse del mantenimiento del hardware, sistema operativo o el almacenamiento subyacente, pero al mismo tiempo, el cliente tiene acceso a un nivel de control y adaptaciones que le permiten satisfacer sus necesidades[70].

La gestión de entregas de software en el caso de SaaS depende, en buena medida, del nivel de configuración y adaptaciones que sea permitido por el proveedor del servicio.

En general, contiene los siguientes elementos

[69] GOOGLE LLC. *Releases*. https://developers.google.com/assistant/console/releases.
[70] PRITHVIRAJ, Sankaran. *Architecting Cloud SaaS software*. p. 5.

- Área de integración: dependiendo de la tecnología utilizada por el proveedor del servicio, se da acceso a un sistema que permita integrar todos los cambios y almacenar la información correspondiente.

- Réplicas del ambiente: para la gestión de cada nueva funcionalidad o ajuste realizado, el mismo debe pasar por múltiples etapas en las que los cambios son aplicados a un ambiente de pruebas similar al de producción.

- Sistemas de promoción entre ambientes: para la migración de los cambios o nuevas funcionalidades entre ambientes, se utilizan sistemas especializados que brindan una capa de abstracción para facilitar la gestión de entregas de software[71].

3.7.4. En premisas

En premisas, es el término utilizado para indicar que un sistema de software es ejecutado utilizando infraestructura que se encuentra bajo el control de la misma empresa. Esto implica que la organización es responsable de todas las capas del sistema.

> La gestión de entregas de software, cuando el mismo es ejecutado en premisas, está estrechamente atada con la administración de la infraestructura sobre la que se implanta el software a entregar, cualquier cambio inesperado en la infraestructura puede tener un efecto negativo en las entregas de software, por lo que, con el paso del tiempo, se han desarrollado las siguientes mejores prácticas[72]:
>
> - Administración de la infraestructura como código: todo cambio a los sistemas de hardware debe ser realizado utilizando software que permita su automatización, registro y auditoría.
> - Pruebas continuas: cada cambio debe ser sometido a pruebas automatizadas que permitan detectar errores en la infraestructura mucho antes de que los mismos lleguen al ambiente de producción.
> - Construir unidades pequeñas: cuando los sistemas son exitosos crecen conforme la demanda aumenta, por lo que es conveniente construir unidades más simples y pequeñas que puedan escalar de forma horizontal.

Es común que para implementar dichas prácticas se utilice una capa de virtualización, instalada sobre el equipo físico administrado por la empresa, lo que le proporciona al equipo de operaciones el nivel de control y flexibilidad requerido.

---

[71] BAHRI, Tameem. *Becoming a Salesforce Certified Technical Architect*. p. 331.
[72] MORRIS, Kief. *Infrastructure as Code*. p. 3.

“El resultado final es una gestión de entregas de software en la que la infraestructura recibe los mismos beneficios que el software”[73].

[73] FINN, Aidan, VREDEVOORT, Hans, LOWNDS, Patrick, et al. *Microsoft Private Cloud Computing*. p. 251.

4. MODELO DE GESTIÓN DE ENTREGAS DE SOFTWARE PARA INCREMENTAR LA PRODUCTIVIDAD DEL EQUIPO DE DESARROLLO DEL PROYECTO DEBIAN GNU/LINUX

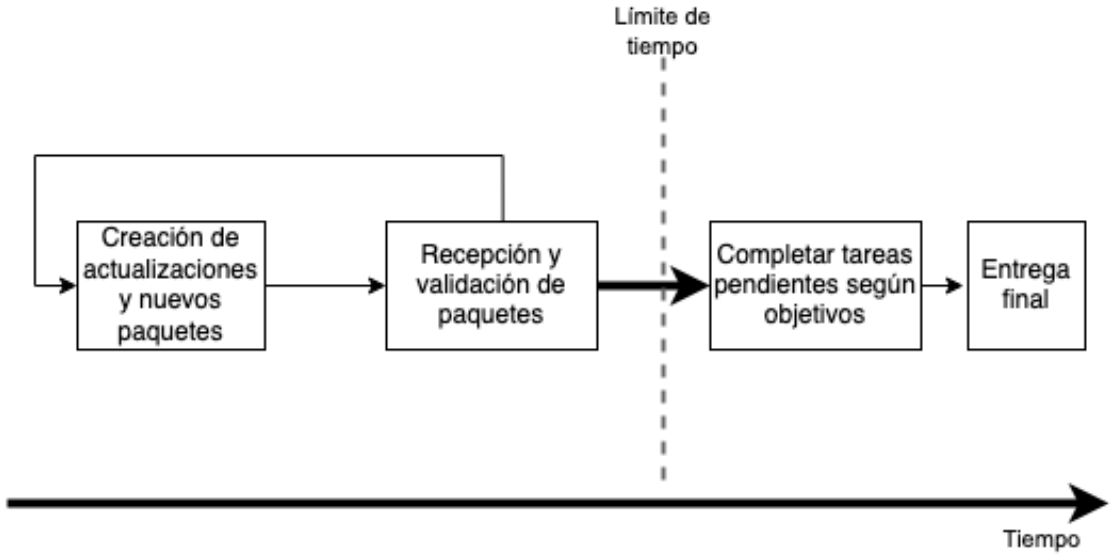
En el presente capítulo se describe el proceso de gestión de entregas de software que actualmente sigue el equipo de desarrollo del proyecto Debian/GNU Linux y luego se describen las oportunidades de mejora que pueden ser aprovechadas por un nuevo modelo de gestión de entregas de software que, combinando los procesos y mejores prácticas expuestas en el capítulo tres, pueda mejorar la productividad en conjunto de todo el equipo de trabajo del proyecto Debian GNU/Linux.

4.1. Proceso actual de gestión de entregas de software

El proceso que sigue actualmente el proyecto Debian GNU/Linux para la gestión de entregas de software está basado en el tiempo y aunque, como se mencionó en la sección 2.5, se tienen otros aspectos que se deben cumplir para completar una nueva versión del sistema operativo, el tiempo es el que determina el inicio, el final y la prioridad de las acciones en el proceso de gestión de entregas[74]. La siguiente figura presenta de manera simplificada el impacto que tiene el tiempo en la gestión de entregas de software:

[74] REICHLE, Meike. *Debian to adopt time-based releases*. https://lwn.net/Articles/344007/.

Figura 22. **Proceso actual basado en el tiempo**

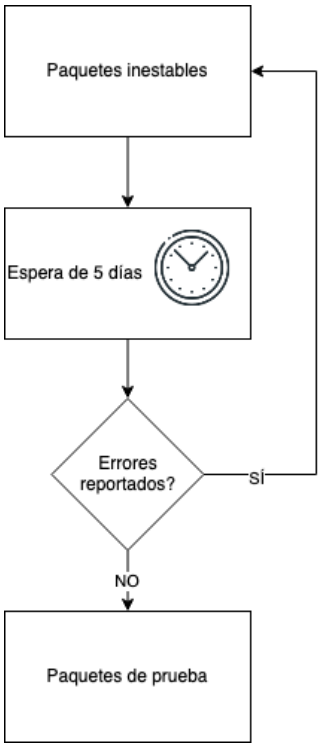


Fuente: elaboración propia, empleando Drawio.

Como se muestra en la figura anterior, el equipo de desarrollo de Debian GNU/Linux inicia cada ciclo de trabajo creando actualizaciones y nuevos paquetes de software de forma repetitiva, hasta que se alcanza el límite de tiempo planificado. Llegado este punto en el tiempo, el equipo pasa a enfocarse en resolver los errores críticos que aún no estén resueltos y completar las tareas que aún estén pendientes para alcanzar los objetivos de la entrega actual. Luego de completar todos los objetivos se procede a realizar la entrega final a los usuarios del sistema operativo GNU/Linux.

Como parte de la validación de los paquetes de software, otro factor que se utiliza es la colaboración de usuarios dispuestos a probar las actualizaciones y nuevos paquetes de software quienes reportan errores en el funcionamiento del software.

Figura 23. **Validación de paquetes**



Fuente: elaboración propia, empleando Drawio.

La figura anterior ilustra el proceso de validación de paquetes de software. Todos los paquetes nuevos y actualizaciones inicialmente forman parte del grupo de paquetes inestables, dicho grupo de paquetes está disponible para que un grupo de usuarios que, de forma voluntaria, hacen uso de estos paquetes de software y, en caso de encontrar errores, envían un reporte de error con los detalles del problema encontrado.

Luego de una espera de cinco días el sistema verifica si existen reportes de errores asociados al paquete de software, si no existen reportes de errores el paquete pasa a formar parte del grupo de paquetes de prueba o de lo contrario el paquete se mantiene en el grupo de paquetes inestables.

“Los paquetes de prueba son los paquetes que finalmente formarán parte de la siguiente versión del sistema operativo Debian GNU/Linux”[75].

En las siguientes secciones se detallan otros elementos que permiten completar los procesos para la gestión de entregas de software en el proyecto Debian GNU/Linux.

4.1.1. Organización actual del equipo de desarrolladores

En el modelo actual de gestión de entregas de software, se pueden identificar dos equipos que juegan un papel crítico en el proyecto Debian GNU/Linux:

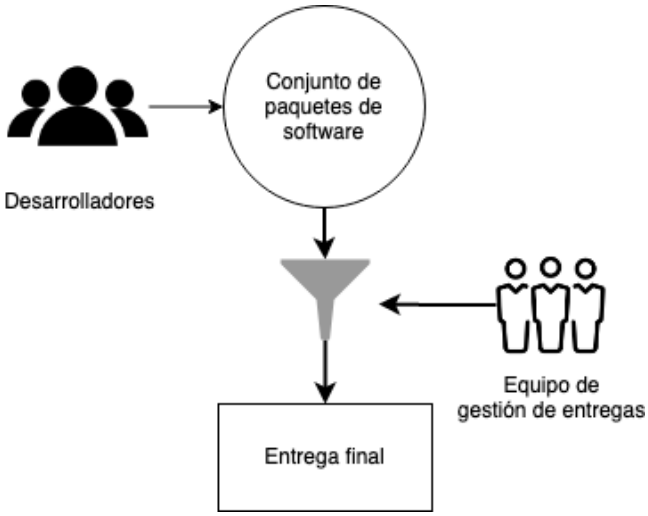
Equipo de gestión de entregas: tiene la máxima autoridad para definir la fecha límite para culminar el ciclo de desarrollo, decidir los objetivos de la siguiente entrega y también decidir si un paquete de software debe ser excluido de la entrega final.

Equipo de desarrollo de paquetes: está constituido por el equipo de voluntarios que se hacen cargo de la creación de actualizaciones y nuevos paquetes de software y son los que se ven directamente influenciados por las decisiones del equipo de gestión de entregas.

La interacción directa entre ambos roles se da sobre todo en la fase final del ciclo de trabajo, sin embargo, como se detalla en las siguientes secciones, el equipo de gestión de entregas también tiene injerencia en la aceptación o rechazo de paquetes nuevos. La siguiente figura presenta de forma conceptual la interacción entre ambos roles:

[75] SPI. *Debian “testing” distribution*. https://www.debian.org/devel/testing.

Figura 24. **Roles en la gestión de entregas**



Fuente: elaboración propia, empleando Drawio.

4.1.2. Flujo de trabajo para colaboradores externos

Existe un flujo de trabajo alternativo que permite a colaboradores externos al proyecto Debian GNU/Linux hacer contribuciones de código. Para llevar a cabo dicho flujo debe ponerse en contacto con un integrante del equipo de desarrollo de Debian GNU/Linux, para que integre las actualizaciones o paquetes nuevos creados por el colaborador externo. El integrante del equipo de desarrollo juega el papel de control de calidad, proponiendo mejoras al colaborador externo, hasta que el paquete nuevo o actualización cumpla con requisitos del proyecto Debian GNU/Linux. La siguiente figura ilustra el flujo de trabajo para un colaborador externo:

Figura 25. **Flujo externo**

Fuente: elaboración propia, empleando Drawio.

4.1.3. Construcción de paquetes

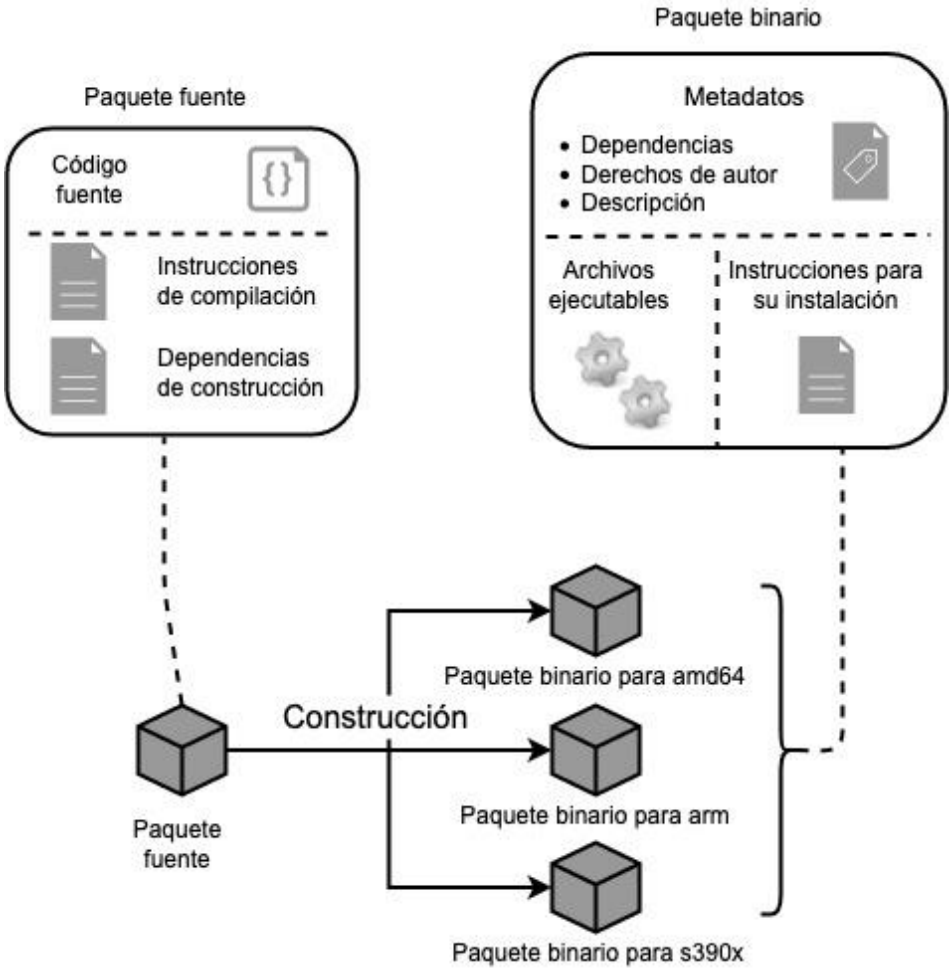
"La unidad básica que forma el sistema operativo Debian GNU/Linux es el paquete de software y existen dos tipos de paquetes de software"[76]:

Paquetes fuente: son los que proveen el código fuente, instrucciones de compilación y metadatos para la construcción del software.

Paquetes binarios: estos son construidos a partir de los paquetes fuente y contienen el software en formato ejecutable para las distintas arquitecturas de hardware, así como la información para su instalación en el sistema operativo.

[76] SPI. *Debian Policy Manual*. https://www.debian.org/doc/debian-policy/index.html.

Figura 26. **Tipos de paquetes**



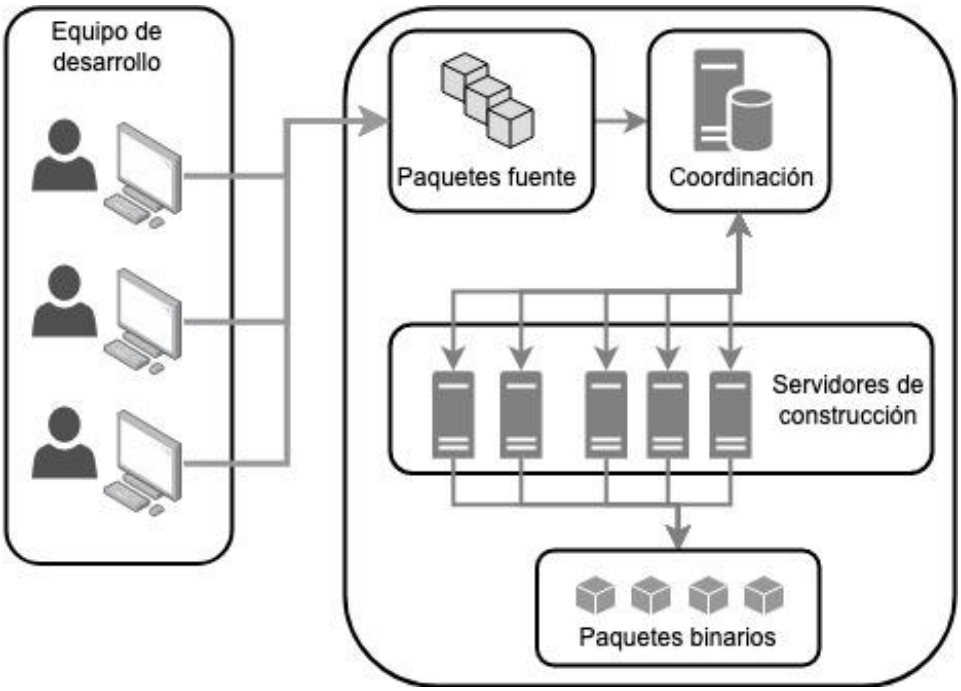
Fuente: elaboración propia, empleando Drawio.

La figura anterior ilustra la relación que entre los paquetes fuente y los paquetes binarios. Gracias al proceso de construcción, un paquete fuente genera múltiples paquetes binarios que contienen los archivos ejecutables para cada una de las arquitecturas de hardware soportadas por el sistema operativo Debian GNU/Linux. En la actualidad el sistema operativo provee soporte para diez arquitecturas de hardware[77].

[77] SPI. *Debian 10 Búster publicado*. https://www.debian.org/News/2019/20190706.es.html.

El equipo de desarrollo se encarga de crear paquetes fuente, los cuales son enviados al sistema de construcción del proyecto Debian GNU/Linux. Debido a que el catálogo de paquetes de software de Debian GNU/Linux incluye más de veinte mil paquetes, el sistema de construcción fue implementado para realizar el trabajo de forma paralela y automática de manera que puede escalar de forma horizontal para aumentar su capacidad.

Figura 27. **Sistema de construcción**



Fuente: elaboración propia, empleando Drawio.

Como se ilustra en la figura anterior, el sistema de construcción está compuesto por una red de servidores de construcción con una coordinación central que identifica los paquetes fuente que necesitan ser procesados para generar los correspondientes paquetes binarios. Al finalizar el proceso, el sistema notifica al responsable del paquete enviando un registro de los resultados[78].

[78] SPI. *Debian autobuilder network*. https://www.debian.org/devel/buildd/.

En cuanto a pruebas, el proceso de construcción contempla principalmente un conjunto limitado de pruebas automáticas que son requeridas, principalmente para la validación del contenido del paquete fuente, cumplimiento de estándares y metadatos. El sistema de construcción también ejecuta las pruebas unitarias que sean incluidas como parte del código fuente y provee la opción a los desarrolladores de ejecutar pruebas instalación y verificación del paquete binario instalándolo automáticamente y validando que el software sea funcional, sin embargo, esto es opcional.

4.1.4. Almacenamiento y distribución de paquetes

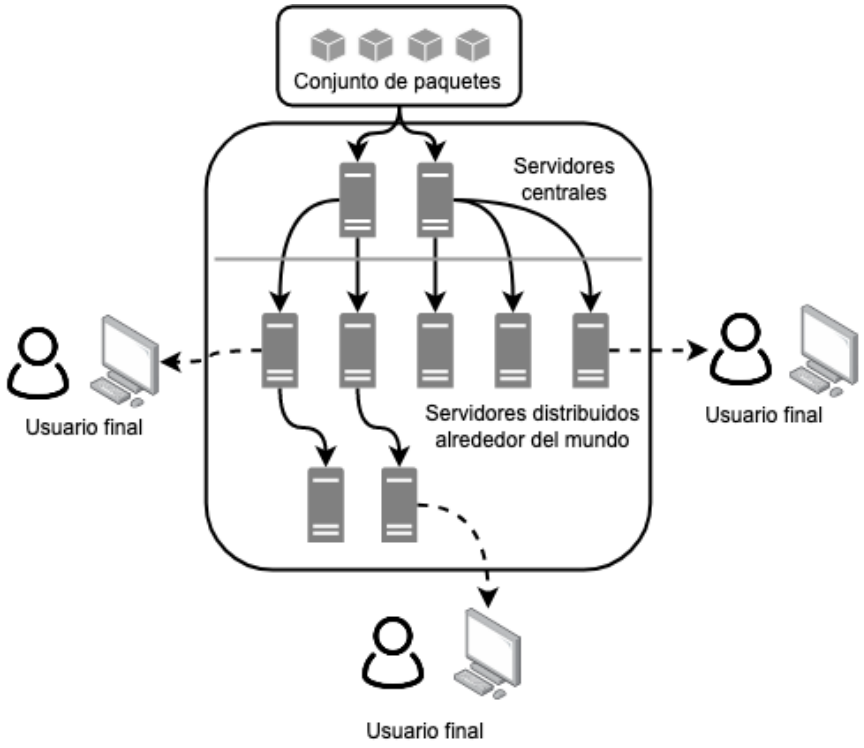
Luego de ser construidos, los distintos paquetes de software que conforman el sistema operativo Debian GNU/Linux, son almacenados en servidores públicos desde los cuales el sistema de administración de paquetes utilizado por los usuarios finales puede obtenerlos e instalarlos. Cada versión del sistema operativo es almacenada en una sección independiente del servidor, de manera que el sistema de administración de paquetes solamente accede a la sección que le corresponde, esto permite que los nuevos paquetes y actualizaciones de la siguiente versión puedan estar almacenados en los servidores públicos, pero estos no son utilizados hasta que se completa la nueva versión del sistema operativo.

Los servidores en los que se almacenan los paquetes de software tienen un sistema de réplicas mediante el cual, toda la información es copiada a múltiples servidores alrededor del mundo[79], lo que permite que los usuarios finales puedan utilizar el servidor que se encuentre más cercano a su ubicación geográfica. Como se ilustra en la siguiente figura, los servidores transmiten la información

[79] SPI. *Debian mirrors*. https://www.debian.org/mirror/sponsors.

unos a otros, lo que evita que los servidores centrales se sobrecarguen y logra que la información se propague rápidamente[80]:

Figura 28. **Sistema de réplicas de distribución**



Fuente: elaboración propia, empleando Drawio.

4.1.5. Entrega de paquetes de software

Como se describe en las secciones anteriores, la construcción de paquetes de software se realiza de forma continua conforme el equipo de desarrollo avanza creando nuevos paquetes y actualizaciones, sin embargo, estos paquetes no son

[80] SPI. *Setting up a Debian archive mirror*. https://www.debian.org/mirror/ftpmirror.

accedidos por los usuarios finales hasta que una nueva versión del sistema operativo es completada.

Para completar la entrega final de una nueva versión del sistema operativo, el equipo de gestión de entregas debe realizar los siguientes pasos[81]:

- Identificar y documentar los errores que no necesitan ser solucionados antes de completar la entrega.
- Coordinar la publicación de la documentación para la nueva versión del sistema operativo.
- Preparar el sitio web del proyecto con toda la información correspondiente a la nueva versión.
- Coordinar la creación y publicación de los discos de instalación.
- Coordinar todos los cambios requeridos en los servidores que distribuyen los paquetes de software.
- Enviar notificaciones a todas las partes interesadas.
- Notificar a la lista de correos de usuarios finales.
- Notificar a medios de comunicación especializados en software de código abierto.
- Notificar a otros proyectos de software que utilizan Debian GNU/Linux como base.

Luego de completados los pasos listados anteriormente, los usuarios proceden a actualizar su sistema operativo utilizando el administrador de paquetes de software, que se encarga de descargar todas las actualizaciones y también le da acceso al usuario para que pueda instalar los nuevos paquetes de software que fueron agregados a Debian GNU/Linux.

4.2. Áreas de oportunidad para el desarrollo de un nuevo modelo

Como resultado del análisis de la información presentada en el capítulo 2 y el proceso utilizado en la actualidad presentado en la sección 4.1, se identificaron múltiples áreas de oportunidad clasificadas como parte de dos temáticas principales:

- Flexibilizar el proceso de entregas

[81] THYKIER, Niels. *Status on the stretch release*. https://lists.debian.org/debian-devel-announce/2017/04/msg00008.html.

- Acciones basadas en el nivel de productividad

Estas áreas de oportunidad tienen el potencial de aumentar la cantidad de paquetes nuevos y actualizaciones entregadas al usuario final. En las siguientes secciones se detallan ambas temáticas.

4.2.1. Flexibilizar el proceso de entregas

El proceso de entregas actual sigue un esquema rígido basado en el tiempo y en el trabajo que se realiza de forma intermitente, sin embargo, si se agregan pasos y requerimientos específicos al proceso para incrementar la consistencia y la detección temprana de errores, permitirá que la entrega de paquetes de software al usuario final pueda seguir esquemas que no dependan de períodos de tiempo limitados.

Las áreas que se enumeran bajo esta temática comprenden aquellas mejoras que permitirán aprovechar en un menor tiempo y de forma consistente el trabajo de los desarrolladores, aumentando la calidad de los paquetes de software.

4.2.1.1. Ejecución de pruebas automáticas

Como se menciona en la descripción del sistema de construcción actual, la mayoría de las pruebas no son obligatorias, por lo que un paquete de software solo requiere tomar en cuenta las pruebas básicas de contenido. El automatizar y convertir en un requerimiento las pruebas, incrementa la calidad de los paquetes de software debido a que se reduce el margen de error humano y se obliga a una mejora continua que junto a la detección temprana de errores

incrementa la productividad. A continuación, se enumeran los aspectos a mejorar:

- Ampliar la cantidad de pruebas de estándares y metadatos requeridas
- Requerir pruebas de componentes
- Realizar pruebas de actualización e instalación
- Requerir pruebas automáticas que validen la funcionalidad del paquete de software

Estos aspectos serán detallados en las secciones 4.3.1 y 4.3.2 del modelo propuesto en el presente capítulo.

4.2.1.2. Validación del usuario final

El proceso actual descrito en la sección 4.1 depende de la colaboración de usuarios en una ventana de tiempo de cinco días para la detección de errores. Esta forma de obtener validación de los usuarios presenta el siguiente riesgo: la ventana de tiempo puede concluir sin que algún voluntario instale y utilice el paquete de software. Lo anterior tiene el efecto final de convertir la validación del usuario en un proceso completamente opcional.

La información que proporcionan los usuarios puede acelerar la detección y solución de problemas. Es por esta razón, que convertir este paso en un requerimiento para todos los paquetes de software, puede proveer una mejora al proceso de desarrollo, aún en el caso en el que el paquete tenga pocos usuarios. En este sentido se identificaron las siguientes oportunidades de mejora:

- Pruebas explicitas: que consiste en mejorar el trabajo de los voluntarios que realizan pruebas, mediante la especificación de una serie de pasos

definidos por el encargado del paquete para la ejecución de la prueba y la descripción de los resultados esperados.

- Validar todos los paquetes: que plantea cambiar a un modelo que provea validación de usuarios para todos los paquetes y no solamente para los más populares.

Los aspectos mencionados anteriormente son tomados en cuenta en el modelo propuesto y se detallan en la sección 4.3.3.

4.2.1.3. Canal de entregas adicional

Actualmente las actualizaciones y paquetes nuevos son entregados siguiendo un único proceso, que se completa exclusivamente cuando la totalidad del sistema operativo está listo para realizar una nueva entrega al usuario final. Sin embargo, al considerar las mejoras detalladas en las secciones anteriores se abre la posibilidad de implementar un canal de entregas adicional que permita entregar actualizaciones y paquetes nuevos a los usuarios finales sin esperas innecesarias.

El incluir pruebas automatizadas y pruebas manuales reduce los errores y aumenta la confianza en el proceso de desarrollo, lo que se puede combinar con un marco para determinar el nivel de riesgo de manera individual para un paquete de software. Estos aspectos son tomados en cuenta para la propuesta de un nuevo modelo que incluye un canal de entregas adicional en la sección 4.3.4.3.

4.2.2. Acciones basadas en el nivel de productividad

Las oportunidades de mejora bajo esta temática cubren los aspectos que tienen la capacidad de proveer al equipo de gestión de entregas del proyecto

Debian GNU/Linux la información para tomar acciones basadas en la productividad del equipo de desarrollo y los objetivos del proyecto, optimizando el uso de recursos y el tiempo invertido en el ciclo de desarrollo.

4.2.2.1. Recolección y difusión de métricas

El proyecto Debian GNU/Linux actualmente almacena un registro detallado de su proceso de desarrollo que puede ser procesado manualmente para calcular múltiples métricas[82], sin embargo, esta información no es una parte activa del proceso de desarrollo. La mejora de este aspecto del proceso de desarrollo de Debian GNU/Linux se puede alcanzar mediante:

- Recolección y procesamiento de datos.
- Difusión de las métricas generadas.

La información que actualmente se recolecta debe ser procesada para que se traduzca en métricas significativas que reflejen los objetivos del proyecto para luego difundir esta información a todos los integrantes del proyecto Debian GNU/Linux de forma continua.

Adicionalmente, cada objetivo específico definido para la siguiente entrega de Debian GNU/Linux debe ser medible periódicamente, de manera que todos los integrantes del proyecto puedan verificar el avance en cualquier momento durante el ciclo de desarrollo. Los aspectos planteados anteriormente son tomados en cuenta en la sección 4.3.4 del modelo propuesto.

[82] SPI. *Ultimate Debian Database*. https://udd.debian.org/.

4.2.2.2. Acciones basadas en métricas

Como se expuso anteriormente, el proveer métricas y objetivos medibles puede brindar al equipo de gestión de entregas del proyecto Debian GNU/Linux la base para tomar acciones efectivas en cada ciclo de desarrollo.

Una revisión periódica del avance completado por el equipo de desarrollo mes a mes puede convertirse en la herramienta más importante del equipo de gestión de entregas del proyecto Debian GNU/Linux. Basándose en esta información, el proyecto puede iniciar inmediatamente el proceso para hacer llegar una nueva versión del sistema operativo a los usuarios finales, sin tener que esperar un plazo de tiempo fijo, lo que se traduce en resultados que aprovechan al máximo la productividad del equipo de desarrollo y maximizan los beneficios recibidos por los usuarios utilizando el menor tiempo posible.

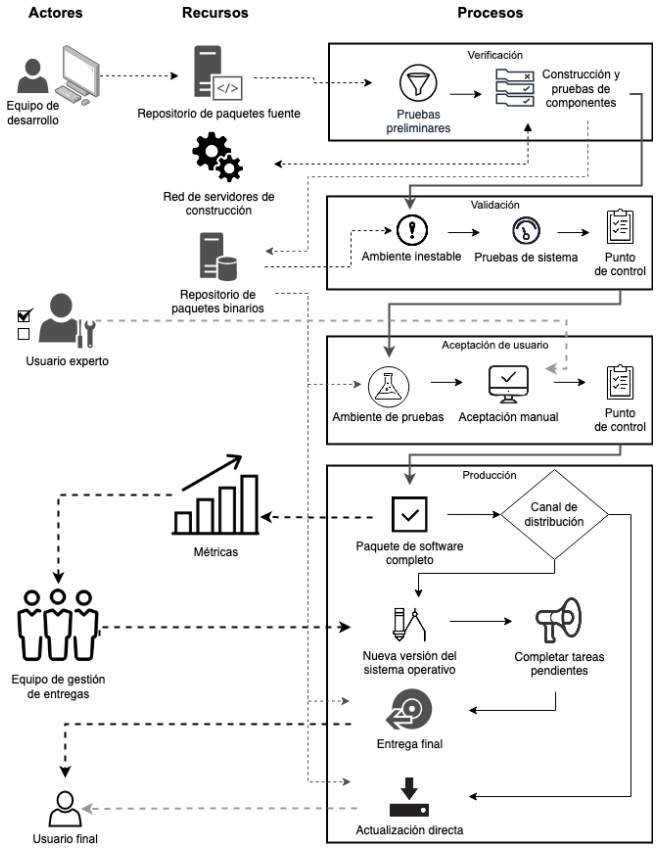
Adicionalmente, para mejorar el flujo de trabajo del equipo de gestión de entregas, es posible recurrir a la evaluación periódica de los resultados, con el objetivo de identificar bajas en la productividad del equipo de desarrollo y basado en esta información trabajar en conjunto para identificar problemas y posibles soluciones que faciliten y hagan más efectivo el trabajo de cada voluntario que participa en el proyecto Debian GNU/Linux. Esta área de oportunidad sirve de base para la sección 4.3.4.1 del modelo propuesto.

A continuación, se presenta el modelo de gestión de entregas de software para el proyecto Debian GNU/Linux que toma en cuenta las áreas de mejora mencionadas previamente.

4.3. Modelo de gestión de entregas de software

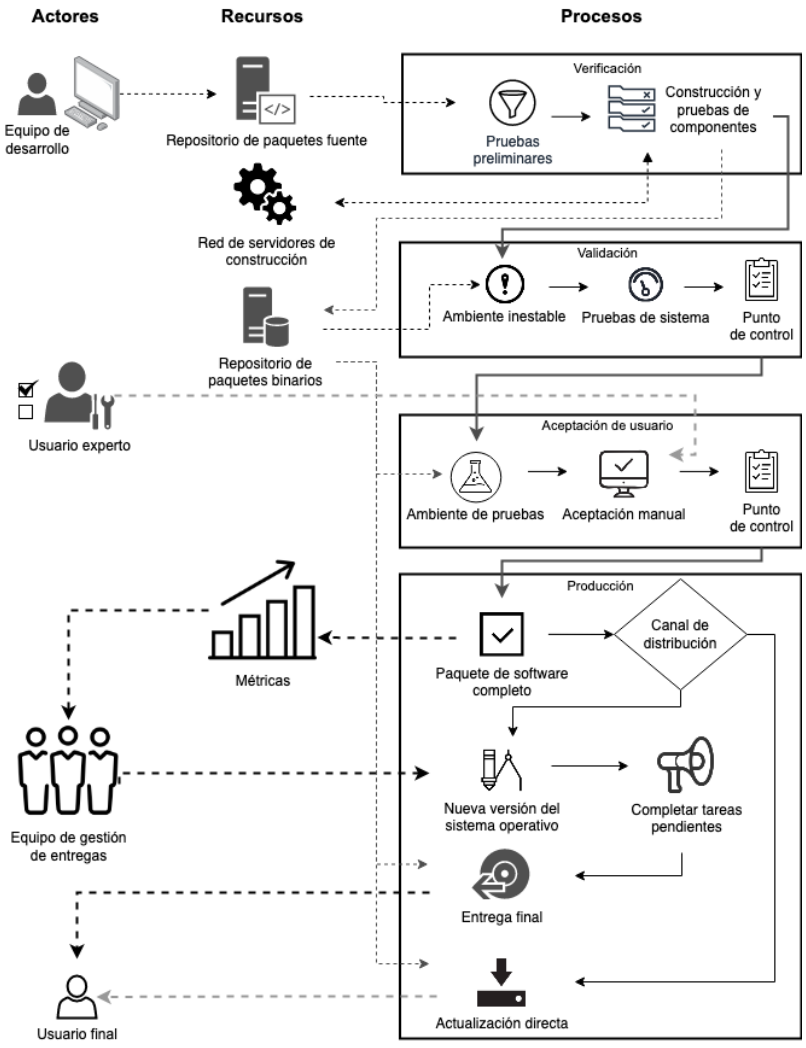
La siguiente figura presenta el esquema general del modelo propuesto para la gestión de entregas de software, el cual será detallado posteriormente:

Figura 29. **Esquema general del modelo**



Fuente: elaboración propia, empleando Drawio.

Figura 30. **Esquema general del modelo**



Fuente: elaboración propia, empleando Drawio.

La figura anterior ilustra los actores, recursos y procesos principales que forman parte del modelo propuesto. Se separan los procesos en cuatro etapas por las que debe pasar todo paquete de software, el cual es desarrollado para formar parte del sistema operativo Debian GNU/Linux:

- Verificación
- Validación
- Aceptación de usuario
- Producción

Estas fases fueron definidas considerando:

- Las mejores prácticas expuestas en el capítulo 3.
- El proceso de gestión de entregas que sigue actualmente el proyecto Debian GNU/Linux.
- Las áreas de mejora expuestas en la sección 4.2.

La relación entre cada fase del modelo propuesto y las mejores prácticas en la gestión de entregas de software se presenta en la siguiente tabla:

Tabla XV. **Fases del modelo propuesto vs. mejores prácticas en la gestión de entregas de software**

Fase del modelo propuesto	Mejores prácticas	Sección	Aspecto
Verificación	DevOps	3.4	- Automatizar cada una de las etapas
	Entregas continuas	3.5	- Con cada entrega se analiza el código de forma automática
Validación	ITIL / ISO 2000	3.1.3.3	- El software es el resultado de un proceso diseñado
	DevOps	3.4	- Automatizar cada una de las etapas
	Gestión de entregas continuas	3.5	- Construcción automática - Pruebas de aceptación automáticas
Aceptación de usuario	ITIL / ISO 2000	3.1.3.4	- Asegurarse que el software ha sido probado
	ANSI/PMI 99-001-2017	3.2.3	- El cliente participa en la ejecución de pruebas - Definición clara de pruebas y resultados esperados
	COBIT 2019	3.3.5	- Ejecución de pruebas definidas con anterioridad - El cliente participa en la ejecución de pruebas
	Gestión de entregas continuas	3.5	- Se realizan pruebas manuales previo a pasar a la siguiente etapa
Producción	ITIL / ISO 2000	3.1.3.1, 3.1.3.2, 3.1.3.5	- Se sigue una definición clara del alcance de la entrega - Se sigue un proceso de entrega definido - Documentación
	ANSI/PMI 99-001-2017	3.2.3	- Se sigue una definición clara del alcance de la entrega - Gestión del tiempo - Gestión de riesgo
	COBIT 2019	3.3.6	- Se sigue un proceso de entrega definido
	Tipo de software ERP	3.7.1	- Plantea el manejo de diferentes tipos de entregas basándose en el contenido y la urgencia

Fuente: elaboración propia, empleando Microsoft Word.

Las columnas presentadas en la tabla anterior representan:

- Fase: nombre de la fase correspondiente del modelo propuesto en este capítulo.
- Mejores prácticas: nombre de las mejores prácticas en la gestión de entregas de software.
- Sección: sección específica del capítulo 3 en el que se expone la información correspondiente.
- Aspecto: en esta columna se presenta de manera corta, el aspecto que relaciona la fase con la mejor práctica en el capítulo 3.

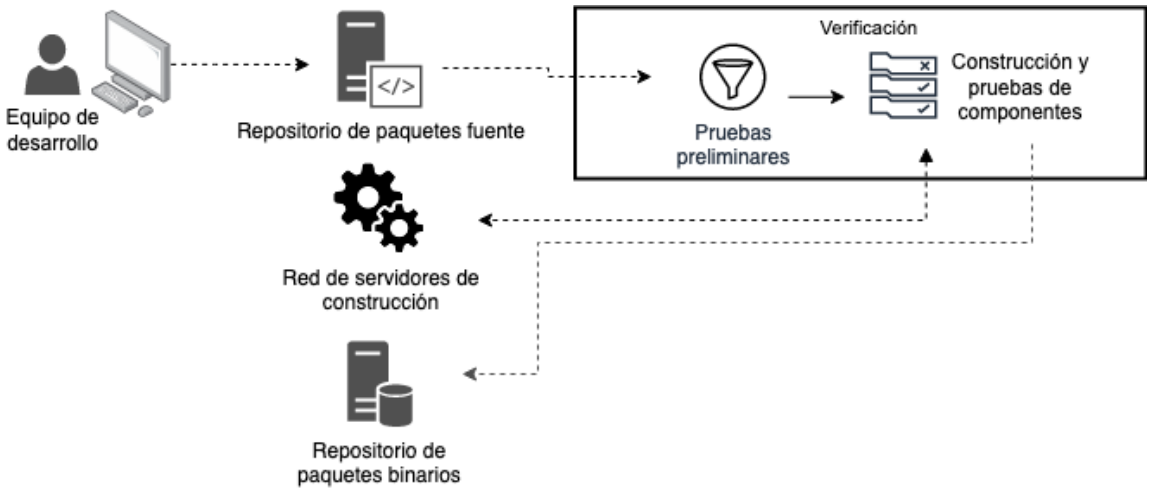
Para obtener más detalles de las mejores prácticas se hace referencia específica a cada una de las secciones del capítulo 3. Como se expuso, las mejores prácticas para la gestión de entregas de software se limitan a describir los aspectos a cubrir de forma general, de manera que estos puedan ser implementados de muchas formas y según las necesidades del proyecto.

Cada etapa del modelo propuesto debe seguirse de forma secuencial completando cada uno de los procesos y requerimientos. En las siguientes secciones se presentan los detalles de cada etapa.

4.3.1. Verificación

La fase de verificación constituye la primera etapa en la que el software entregado por el equipo de desarrollo es probado y transformado en paquetes binarios. A continuación, se presenta el esquema para la fase de verificación, el cual será detallado en las siguientes secciones:

Figura 31. **Esquema de verificación**



Fuente: elaboración propia, empleando Drawio.

La fase de verificación inicia cuando un integrante del equipo de desarrollo entrega un nuevo paquete fuente, dicha entrega se realiza utilizando el repositorio de paquetes fuente, el cual consiste en un servidor de almacenamiento que al detectar la recepción de nuevos paquetes de software inicia el proceso de validación de forma automática.

La figura anterior presenta como primer paso a realizar la ejecución de pruebas preliminares, las cuales tienen el objetivo de verificar que todo paquete de software cumple con los requisitos y estándares de calidad esperados. Se consideran dos tipos de pruebas a ejecutar:

- Verificación de estándares
- Verificación de metadatos

La verificación de estándares comprende la comprobación de la estructura del paquete, los tipos de archivos, mejores prácticas de seguridad y el sistema de

construcción utilizado. El incrementar la cantidad y el esfuerzo invertido en este tipo de pruebas permitirá mejorar el cumplimiento con la normativa de Debian GNU/Linux y fomenta la participación del equipo de desarrollo en la definición de dicha normativa[83].

Los metadatos que son agregados a los paquetes de software son el resultado directo del esfuerzo de los desarrolladores y su validación suele quedar en segundo plano. El presente modelo propone la integración de la validación de los metadatos como un requerimiento preliminar en el procesamiento de nuevos paquetes.

La herramienta que es tradicionalmente utilizada para la validación de metadatos y estándares es el software conocido como Lintian[84], la cual debe pasar a ser un requisito en el proceso de construcción, ejecutando todas las pruebas de validación que provee la herramienta. A continuación, se describen un extracto de pruebas que forman parte del conjunto de validaciones que provee la herramienta:

[83] SPI. *Debian Policy Manual*. https://www.debian.org/doc/debian-policy/index.html.
[84] SPI*. Lintian Reports.* 2020, https://lintian.debian.org/.

Tabla XVI. **Ejemplos de pruebas de estándares y metadatos definidos en la herramienta Lintian**

Nombre de la prueba	Tipo	Descripción
“privacy-breach-uses-embedded-file”	Estándares	Verifica si el paquete viola los estándares de privacidad al contactar un sitio externo
“library-in-root-and-usr”	Estándares	Verifica que las librerías sean instaladas en el directorio correcto del sistema operativo
“wrong-bug-number-in-closes”	Metadatos	Verifica que los números de reportes de error presentes en los metadatos sean correctos
“uploader-name-missing”	Metadatos	Verifica que el valor en el campo llamado *uploader* tenga el valor correcto

Fuente: SPI. *Lintian reports*. lintian.debian.org. https://lintian.debian.org/. Consulta: 4 de abril de 2021.

4.3.1.1. Construcción y pruebas de componentes

El siguiente proceso comprende la construcción y pruebas de componentes, que involucra la compilación del código fuente a código binario, la generación de paquetes binarios y la ejecución de pruebas de componentes. Dichas pruebas son las encargadas de verificar la funcionalidad de un paquete de software de

forma aislada a nivel de código y son ejecutadas inmediatamente luego de compilar el código fuente.

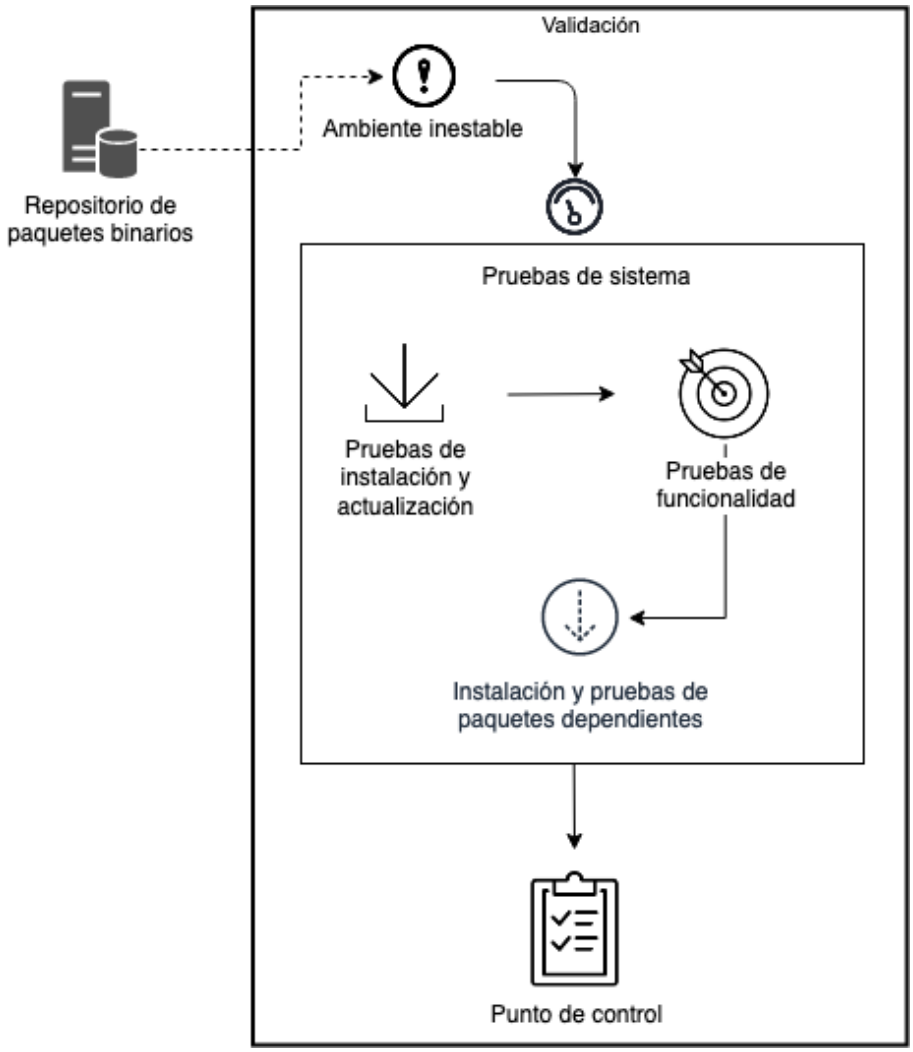
Los leguajes de programación que son interpretados y no requieren ningún tipo de compilación, como por ejemplo Perl o PHP, pasan directamente a la ejecución de sus respectivas pruebas de componentes, las cuales según el presente modelo, son requeridas para todos los paquetes de software sin importar el lenguaje de programación que sea utilizado.

El principal recurso a cargo de la ejecución de este proceso es la red de servidores de construcción, la cual fue descrita en la sección 4.1.3. Cuando el proceso se completa de forma exitosa el resultado es almacenado en el repositorio de paquetes binarios, para luego proceder a la fase de validación.

4.3.2. Validación

La fase de validación se encarga de la detección automática de errores introducidos con cada cambio a un paquete de software, el flujo que a continuación se presenta, establece los procesos a seguir durante esta fase:

Figura 32. **Esquema de validación**



Fuente: elaboración propia, empleando Drawio.

Como muestra el esquema anterior, la fase de validación consta de varios procesos, los cuales inician con la migración del paquete al ambiente inestable. Las pruebas de sistema se llevan a cabo mediante la instalación del paquete de forma automática para la ejecución de pruebas que verifiquen su funcionamiento en conjunto con sus dependencias de la siguiente manera:

- Pruebas de instalación y actualización del paquete.
- Pruebas de funcionalidad del paquete.
- Pruebas de funcionalidad de los paquetes que dependen del paquete actualizado.

Para su ejecución se deben utilizar las siguientes herramientas:

- "Piuparts: permite la ejecución de pruebas de instalación y actualización en un sistema de archivos virtual"[85].

- "Autopkgtest: se encarga de la ejecución de pruebas de funcionalidad definidas por el equipo de desarrollo de Debian GNU/Linux"[86].

La última etapa de la fase de validación es el paso por el punto de control, en el cual todo paquete de software deberá cumplir con los siguientes requisitos para poder avanzar a la siguiente fase:

- Todas las pruebas deben haberse ejecutado obteniendo un resultado satisfactorio.
- Todas las dependencias deben haber ejecutado sus pruebas correspondientes y haber obtenido un resultado satisfactorio.

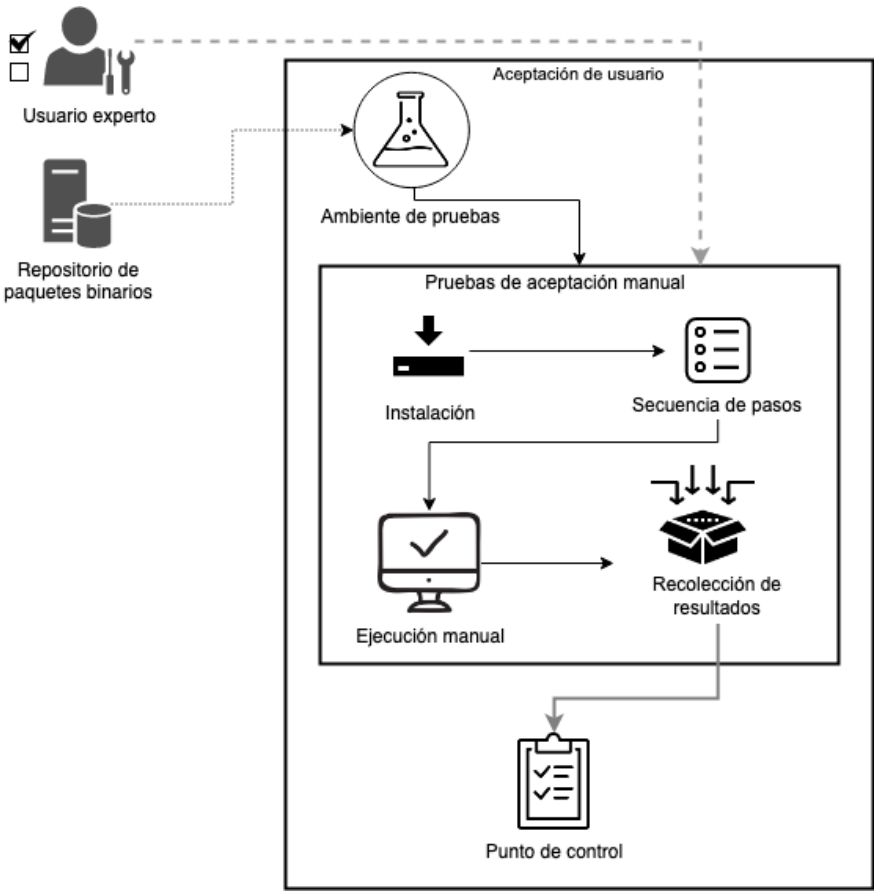
4.3.3. Aceptación de usuario

La fase de aceptación de usuario consiste en el conjunto de procesos, que junto con la participación de múltiples voluntarios, permite la validación de la

[85] WIRZENIUS, Lars. *About piuparts.debian.org*. https://piuparts.debian.org/.
[86] MARTIN, Pitt. *Autopkgtest - Running tests*. https://people.debian.org/~mpitt/autopkgtest/README.running-tests.html.

funcionalidad provista por los paquetes de software. El siguiente esquema presenta el flujo general para la fase de aceptación de usuario, el cual será detallado en esta sección:

Figura 33. **Esquema de aceptación**



Fuente: elaboración propia, empleando Drawio.

Esta fase tiene como actor principal al usuario experto, el cual es un voluntario que colabora realizando pruebas a los paquetes de software nuevos o actualizaciones y reportando los resultados. Para todo paquete que se prueba se busca:

- Verificar que el paquete de software puede ser ejecutado por un usuario
- Comprobar que el paquete de software provee la funcionalidad esperada
- Detectar errores que no fueron detectados por las pruebas automáticas

El sistema de gestión de paquetes de software debe encargarse de instalar las nuevas versiones de paquetes de software y presentar al usuario la descripción de la secuencia de pasos a realizar para completar la prueba, así como el resultado esperado. La secuencia de pasos a realizar debe contener las siguientes secciones:

- Nombre: el cual permite hacer referencia a la prueba a realizar.
- Descripción: este campo describe el propósito de la prueba a realizar.
- Estado inicial: describe las condiciones iniciales de la prueba.
- Acciones por realizar: describe paso a paso las acciones que el usuario debe realizar.
- Resultado esperado: una descripción del resultado esperado luego de completar todos los pasos a realizar.

En el apéndice 1 se presenta un ejemplo de una prueba de aceptación de usuario siguiendo el formato descrito anteriormente, el cual está basado en el esquema propuesto por la autora Van Goethem[87].

El sistema de gestión de paquetes solicitará al usuario proveer los resultados de las pruebas realizadas, recolectando la siguiente información:

[87] HAMBLING, Brian; VAN GOETHEM, Pauline. *User acceptance testing: a step-by-step guide*. p. 144.

Resultado de la prueba: satisfactorio o insatisfactorio, dependiendo de la comparación entre el resultado esperado y el resultado obtenido al ejecutar la prueba.

Comportamiento actual: en caso de que el resultado de la prueba sea insatisfactorio, se solicitará al usuario una descripción del comportamiento erróneo.

Comentario adicional: opcionalmente el usuario puede agregar comentarios que puedan ayudar a mejorar la funcionalidad del software.

El apéndice 2 presenta un ejemplo de un reporte con la información especificada anteriormente, luego de la ejecución de una prueba.

La última etapa de la fase de aceptación es el paso por el punto de control, en el cual todo paquete de software deberá cumplir con los siguientes requisitos para poder avanzar a la siguiente fase:

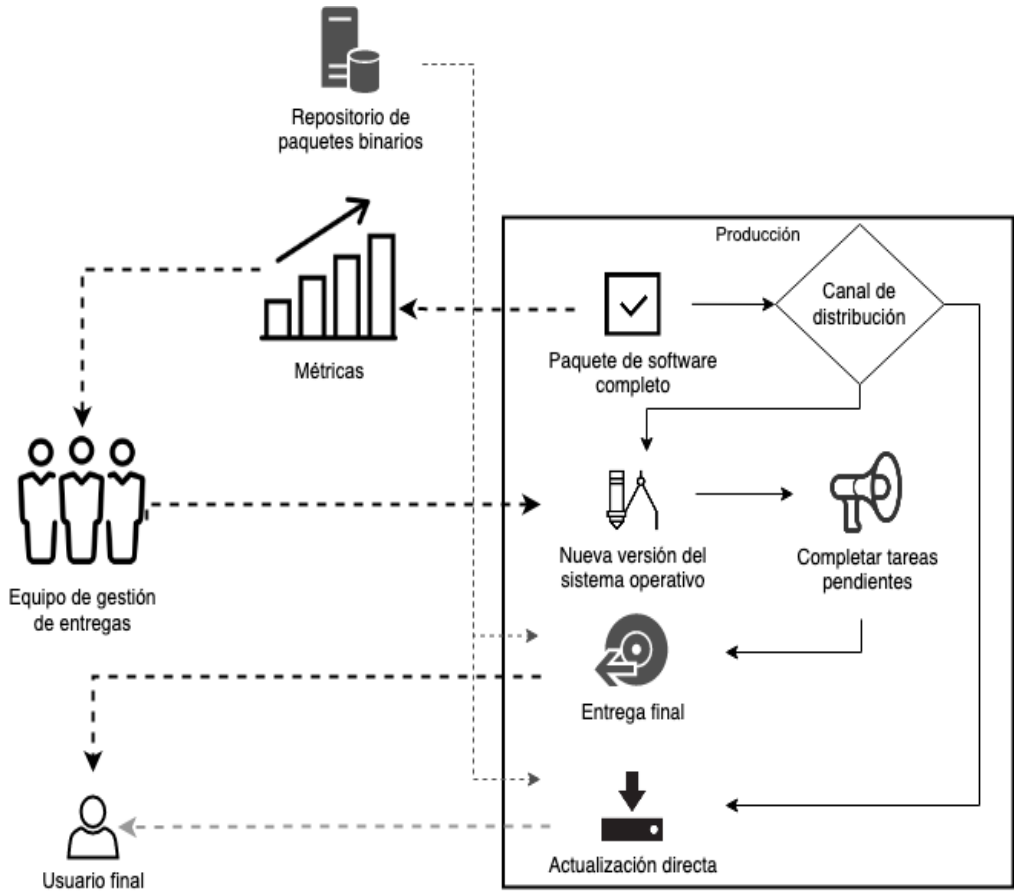
- Haber obtenido por lo menos los resultados de una prueba
- Todas las pruebas reportadas deben tener un resultado satisfactorio

4.3.4. Producción

La fase de producción es la etapa final del modelo de gestión de entregas de software, la cual contempla los mecanismos para dar seguimiento a la productividad del equipo de desarrollo y completar la entrega de nuevas versiones del sistema operativo a los usuarios. A continuación, se presenta el

esquema del flujo general para la fase de producción, la cual será detallada en la presente sección:

Figura 34. **Esquema de fase de producción**



Fuente: elaboración propia, empleando Drawio.

La figura anterior ilustra el flujo a seguir durante la fase de producción, la cual constituye la última fase del modelo para la gestión de entregas de software. Esta fase contempla tres procesos principales:

- Seguimiento de métricas

- Entrega final
- Actualizaciones directas

4.3.4.1. Seguimiento de métricas

Las métricas constituyen el principal criterio para decidir el mejor momento para realizar la siguiente entrega de una nueva versión del sistema operativo. El principal actor de este proceso es el equipo de gestión de entregas, el cual dará seguimiento al avance de los objetivos de la entrega.

Cada paquete que alcanza la fase de producción pasa a ser parte del registro de métricas del proyecto para el ciclo de desarrollo actual. El equipo de gestión de entregas del proyecto Debian GNU/Linux se encargará de evaluar de forman mensual el trabajo realizado de acuerdo con las métricas. Con base en los resultados obtenidos de los ciclos de desarrollo de 2007 a 2019 presentados en la sección 2.3, se definen como principales objetivos:

Completar actualizaciones mayores para el 40 % de los paquetes incluidos en la versión anterior de Debian GNU/Linux. El apéndice 3 presenta la fórmula a utilizar para realizar este cálculo.

Completar un 25 % de paquetes nuevos en comparación con la versión anterior de Debian GNU/Linux. El apéndice 4 presenta la fórmula para realizar este cálculo.

Llegado el punto en el que ambos objetivos sean alcanzados, el equipo de gestión de entregas iniciará el proceso para la publicación de una nueva versión del sistema operativo Debian GNU/Linux, el cual es presentado en la siguiente sección.

Adicionalmente el equipo de gestión de entregas de Debian GNU/Linux tomará en cuenta las siguientes métricas para optimizar el nivel de productividad del equipo de desarrollo de software:

- Producción mensual de paquetes nuevos: mayor o igual a 100 paquetes, ver apéndice 5.
- Producción mensual de actualizaciones mayores: mayor o igual a 100 paquetes, ver apéndice 6.
- Cantidad mensual de errores reportados: menor a 150, ver apéndice 7.

Las cantidades elegidas como métricas de productividad están basadas en los datos históricos que fueron expuestos en la sección 2.3.2 y 2.4 que comprenden el período del 2007 al 2019.

En los casos en los que se detecte una baja en la productividad el equipo de gestión de entregas procederá a:

- Identificar la causa de la baja de productividad y publicar los hallazgos.
- Trabajar con el equipo de desarrollo para proponer posibles mejoras.
- Identificar el impacto de la baja de productividad en el ciclo de desarrollo actual.
- Ajustar los objetivos en caso de ser necesario.

4.3.4.2. Entrega final

El proceso de entrega final consiste en una serie de pasos que permite la entrega de todos los paquetes de software que han alcanzado de forma exitosa la fase de producción integrando la nueva versión del sistema operativo Debian

GNU/Linux. El equipo de gestión de entregas de software tiene la máxima autoridad y control sobre este proceso.

El primer paso por realizar consiste en establecer el listado de paquetes de software que alcanzaron exitosamente la fase de producción y que por lo tanto formarán parte de la nueva versión del sistema operativo Debian GNU/Linux. El apéndice 8 presenta una plantilla para la elaboración de dicho listado, la cual debe ser completada obteniendo la información directamente desde el repositorio de paquetes binarios.

El siguiente paso consiste en la creación y publicación de los discos de instalación, utilizando como referencia el listado de paquetes que forma parte de la nueva versión del sistema operativo, además de lo anterior se debe publicar la documentación correspondiente en el sitio web del proyecto Debian GNU/Linux.

La sección 4.1.4 describe la red réplicas de distribución. El equipo de gestión de entregas se encargará de verificar que todos los paquetes de software incluidos en la nueva versión de Debian GNU/Linux se encuentren disponibles en la red de réplicas de distribución, de manera que los usuarios existentes tengan acceso a actualizar directamente el sistema operativo sin tener que recurrir a los discos de instalación.

Finalmente el equipo de gestión de entregas enviará una notificación mediante correo electrónico a todas las partes interesadas, indicando que la entrega final de la nueva versión del sistema operativo Debian GNU/Linux está disponible. Los pasos a seguir son:

- “Notificar a la lista de correos de usuarios de Debian GNU/Linux”[88]
- Notificar a medios de comunicación especializados en software de código abierto:
- “Distrowatch: sitio web dedicado a la publicación de noticias de sistemas operativos de tipo UNIX de código abierto”[89]
- “LWN: revista en línea especializada en software de código abierto”[90]
- “Slashdot: sitio web de noticias orientado a tecnología”[91]
- “Notificar a otros proyectos de software que utilizan Debian GNU/Linux como base”[92]

4.3.4.3. Actualizaciones directas

El presente modelo contempla la creación de un nuevo canal de entregas de paquetes de software adicional. Este nuevo canal de entregas permitirá el realizar actualizaciones mayores y poner a disposición de los usuarios paquetes nuevos de forma inmediata, siempre y cuando el paquete de software cumpla con los siguientes criterios:

El paquete de software ha alcanzado la fase de producción, esto significa que ha superado satisfactoriamente todas las fases anteriores incluyendo la aceptación manual de usuario.

El paquete está clasificado con un riesgo de nivel bajo, lo que significa que el impacto de actualizar dicho paquete en el sistema operativo del usuario es relativamente bajo. Para la estimación de dicho riesgo se utiliza la información de

[88] SPI. *Debian mailing list for debian-user*. https://lists.debian.org/debian-user/.
[89] ATEA ATAROA LIMITED. *DistroWatch*. https://distrowatch.com/.
[90] EKLEKTIX INC. *LWN.net contact information*. https://lwn.net/op/FAQ.lwn#contact.
[91] SLASHDOT MEDIA. *Slashdot news*. https://slashdot.org.
[92] SPI*. Debian mailing list for derivatives.* https://lists.debian.org/debian-derivatives/.

dependencias que forma parte de cada paquete de software de la siguiente manera:

- Riesgo bajo: cero paquetes dependientes.
- Riesgo medio: de uno a cinco paquetes dependientes.
- Riesgo alto: cinco o más paquetes dependientes.

Es decir, que cuando un paquete de software alcanza la fase de producción, su nivel de riesgo será evaluado de forma automática y será asignado al canal de entregas correspondiente:

Paquetes de riesgo medio y alto serán agregados a la siguiente entrega final

Paquetes de riesgo bajo serán puestos a disposición de los usuarios de forma inmediata.

"Para automatizar este proceso, se deberán hacer cambios al software que administra el repositorio de paquetes binarios llamado Britney"[93], el cual deberá encargarse de evaluar el riesgo y hacer los cambios para que el gestor de paquetes del sistema operativo pueda dar acceso a las actualizaciones y paquetes nuevos de manera inmediata a los usuarios de Debian GNU/Linux.

4.4. Consideraciones para la implementación del modelo

Dada la naturaleza de la organización del proyecto Debian GNU/Linux, la cual está integrada completamente por voluntarios que donan su tiempo y se

[93] THYKIER, Niels. *What is britney and britney2*. https://release.debian.org/doc/britney/what-is-britney.html.

encuentran distribuidos en múltiples regiones del mundo, para la implementación del modelo propuesto se deben tomar en cuenta las siguientes consideraciones:

Difundir el modelo propuesto como un primer paso, de manera que los integrantes de la organización puedan dar sus opiniones y discutir los distintos aspectos que abarca el modelo. Para ello, el escenario ideal es la reunión anual de desarrolladores, debido a que esta reunión es presencial y permite una interacción fluida.

Tomando en consideración que el equipo más afectado por el modelo propuesto es el equipo de gestión de entregas del proyecto Debian GNU/Linux, se debe buscar su aprobación, para posteriormente elaborar un plan de implementación por etapas, de manera que se dé una transición gradual, reduciendo la fricción y la resistencia al cambio de la mejor manera posible.

Establecer los canales de comunicación para notificar cada hito alcanzado en el proceso de implementación y manteniendo la ventana abierta para escuchar las opiniones del equipo de desarrollo y posibles efectos que pudieran surgir durante el proceso implementación, pues para el proyecto Debian GNU/Linux la inclusión es uno de los valores más importantes.

CONCLUSIONES

1. Al analizar la organización interna del proyecto Debian GNU/Linux se pudo determinar que el establecer procesos, roles, responsabilidades y estándares ha jugado un papel crucial en la evolución del proyecto, permitiendo el crecimiento constante tanto del equipo de desarrollo como de la cantidad de paquetes de software producidos. Sin embargo, al profundizar en la metodología de trabajo, se encontró una carencia de medición objetiva del desempeño del equipo de desarrollo de software que permita la detección temprana de problemas para así optimizar los resultados que se obtienen en cada ciclo de trabajo.

2. Los registros de los paquetes de software producidos por el equipo de desarrollo del proyecto Debian GNU/Linux durante los últimos diez años, luego de ser agrupados y analizados, permitieron identificar una brecha de hasta un 38 % entre la capacidad de producción y el resultado entregado al usuario final. Los datos mostraron que la capacidad de producción del equipo de desarrollo de software no es aprovechada, debido a que el modelo de gestión de entregas de software actual impone tiempos de espera arbitrarios, considerando el promedio de 24 meses de cada ciclo de desarrollo, se evidenció que carece de flexibilidad, limitándose a hacer estrictamente entregas simultaneas de las nuevas funcionalidades provistas por los paquetes nuevos y actualizaciones mayores.

3. Las mejores prácticas para la gestión de entregas de software, remarcan la importancia de la inversión en infraestructura y en la definición de procesos que permitan la detección temprana de errores y la mejora

continua para incrementar la productividad y calidad de los resultados del proceso de desarrollo de software.

Entre las prácticas analizadas destacan: ITIL, la gestión de entregas continuas y COBIT, debido a que establecen prácticas puntuales que pueden ser aplicadas con la flexibilidad requerida por distintos tipos de proyectos, sin importar el número de integrantes o su ubicación geográfica. Por otra parte, las prácticas observadas en tipos de software específicos como ERP o aplicaciones móviles, contienen patrones que pueden ser reutilizados, aunque en su mayoría dependen de infraestructura especializada por lo que su utilidad es limitada.

4. El modelo de gestión de entregas de software diseñado para el equipo de desarrollo del proyecto Debian GNU/Linux permite optimizar de forma proactiva los resultados obtenidos, incrementando el valor del software entregado a los usuarios del sistema operativo Debian GNU/Linux y reduciendo la duración de los ciclos de desarrollo. El modelo establece procesos, flujos de trabajo y métricas para la evaluación del desempeño, los cuales pretenden mejorar la gestión de entregas de software.

RECOMENDACIONES

1. Motivar a los equipos de desarrollo de software de las organizaciones no lucrativas de Guatemala para que participen en el desarrollo del proyecto Debian/GNU Linux considerando que se pueden beneficiar mutuamente al distribuir software siguiendo su metodología de trabajo e incrementando el catálogo de paquetes de software disponible.

2. Analizar mensualmente los registros del equipo de desarrollo de software del proyecto Debian GNU/Linux con la finalidad de medir la cantidad de paquetes completados e identificar nuevos factores que puedan afectar la productividad para tomar acciones correctivas e implementar posibles mejoras que agilicen el proceso de desarrollo de software.

3. Invertir los recursos necesarios para la creación de un modelo de gestión de entregas de software adaptado a las necesidades de cada proyecto, tomando en cuenta las mejores prácticas establecidas por ITIL, COBIT y la gestión de entregas continuas, evaluando constantemente alternativas que les permita aumentar su productividad.

4. Implementar el modelo de gestión de entregas de software propuesto, el cual permitirá incrementar la productividad del equipo de desarrollo de software del proyecto Debian GNU/Linux debido a que provee un flujo de entregas flexible, así como un proceso para la optimización del tiempo utilizado en cada ciclo de desarrollo, que se basa en métricas y evaluaciones de desempeño continuas.

BIBLIOGRAFÍA

1. AGUILH, Franck. *Understanding software release management*. *Proceedings of the Project Management Institute Annual Seminars & Symposium*. 1a ed. Estados Unidos: Project Management Institute, 2000. 460 p.

2. ARORA, Tarun. *Microsoft team foundation server 2015 cookbook*. India: Packt Publishing, 2016. 315 p.

3. Atea Ataroa Limited. *DistroWatch.* [en línea]. <https://distrowatch.com/>. [Consulta: 4 de abril de 2021].

4. _____. *DistroWatch Page Hit Ranking.* [en línea]. <https://distrowatch.com/dwres.php?resource=popularity>. [Consulta: 4 de enero de 2021].

5. BAHRI, Tameem. *Becoming a Salesforce Certified Technical Architect*. 1a ed. Reino Unido: Packt Publishing, 2021. 628 p.

6. BALGROSKY, Jean A. *Essentials of health information systems and technology*. 1a ed. Estados Unidos: Jones & Bartlett Learning, 2015. 281 p.

7. BON, Jan van. *ISO/IEC 20000 An Introduction*. 1a ed. Países Bajos: Van Haren, 2008. 226 p.

8. COBIT. *Control Objectives for Information Technologies ISACA.* [en línea]. <https://www.isaca.org/resources/cobit>. [Consulta: 21 de febrero de 2021].

9. COLEMAN, E. Gabriella. *Three Ethical Moments in Debian*. 1a ed. Estados Unidos: Rutgers University, 2005. 77 p.

10. _____. *Debian History Roundtable Discussion.* [en línea]. <https://gabriella coleman.org/debian-history-roundtable-discussion/>. [Consulta: 29 de noviembre de 2020].

11. EKLEKTIX INC. *LWN.net contact information.* [en línea]. <https://lwn.net/op/FAQ.lwn#contact>. [Consulta: 4 de abril de 2021].

12. FARLEY, Dave; HUMBLE, Jez. *Continuous delivery: reliable software releases through build, test, and deployment automation.* 1a ed. Estados Unidos: Addison-Wesley Professional, 2010. 512 p.

13. FAYER, Leon. *DevOps and Business*. 1a ed. Estados Unidos: O'Reilly, 2019. 77 p.

14. FINN, Aidan et al. *Microsoft Private Cloud Computing*. 1a ed. Estados Unidos: Wiley, 2012. 408 p.

15. GARBEE, Bdale et al. *A Brief History of Debian.* [en línea]. <https://www.debian.org/doc/manuals/project-history/>. [Consulta: 5 de diciembre de 2020].

16. GOOGLE LLC. *Google Trends.* [en línea]. <https://trends.google.com/>. [Consulta: 4 de enero de 2021].

17. _____. *Releases.* [en línea]. <https://developers.google.com/assistant/console/releases>. [Consulta: 3 de marzo de 2021].

18. HAMBLING, Brian; VAN GOETHEM, Pauline. *User acceptance testing: a step-by-step guide*. 1a ed. Reino Unido: BCS, 2013. 226 p.

19. HUNNEBECK, Lou. *Key element guide ITIL service design*. London: TSO, 2012. ISBN 978-0-11-331361-7.

20. IMDB INC. *Perfil de Bruce Perens.* [en línea]. <http://www.imdb.com/name/nm0673302/>. [Consulta: 5 de diciembre de 2020].

21. International Organization for Standardization. *ISO/IEC 20000-1:2018.* [en línea]. <https://www.iso.org/standard/70636.html>. [Consulta: 15 de febrero de 2021].

22. ISACA. *COBIT 2019 Framework Governance and Management Objectives*. 1a ed. Estados Unidos: ISACA, 2019. 302 p.

23. JACKSON, Ian; BENHAM, Darren. *Debian BTS.* [en línea]. <https://www.debian.org/Bugs/server-control.es.html>. [Consulta: 29 de noviembre de 2020].

24. _____. *Debian BTS - developer info.* [en línea]. <https://www.debian.org/Bugs/Developer#severities>. [Consulta: 10 de enero de 2021].

25. KALAIMANI, Jayaraman. *SAP project management pitfalls*. 1a ed. Estados Unidos: Apress, 2016. 325 p.

26. KHOMH, Foutse et al. *Do faster releases improve software quality? An empirical case study of Mozilla Firefox*. 9a ed. Suiza: IEEE, 2012. 254 p.

27. KOTLER, Philip (ed.). *Principles of marketing*. 4a ed. Reino Unido: Prentice Hall, 2005. 992 p.

28. KRAFFT, Martin F. *The Debian system: concepts and techniques*. 1a ed. Alemania: Open Source Press, 2005. 605 p.

29. KUNAS, Michael. *Implementing Service Quality based on ISO/IEC 20000: A Management Guide*. 3a ed. Reino Unido: IT Governance Publishing, 2012. 109 p.

30. LAPORTE, Claude Y.; APRIL, Alain. *Software quality assurance*. 1a ed. Estados Unidos: Wiley, 2018. 624 p.

31. LEVSEN, Holger et al. *Debian Developer's Reference.* [en línea]. <https://www.debian.org/doc/manuals/developers-reference/index.en.html>. [Consulta: 29 de noviembre de 2020].

32. MARTIN, Pitt. *Autopkgtest - Running tests.* [en línea]. <https://people.debian.org/~mpitt/autopkgtest/README.running-tests.html>. [Consulta: 23 de marzo de 2021].

33. MICHLMAYR, Martin. *Quality Improvement in Volunteer Free and Open Source Software Projects – Exploring the Impact of Release Management*. 1a ed. Reino Unido: Cambridge University, 2007. 212 p.

34. MICROSOFT. *¿Qué es DevOps?* [en línea]. <https://azure.microsoft.com/es-mx/overview/what-is-devops/>. [Consulta: 28 de febrero de 2021].

35. MOHAMED, Samer I. *Software release management evolution-comparative analysis across agile and DevOps continuous delivery*. *International Journal of Advanced Engineering Research and Science*. 3a ed. India: AI Publications, 2016. 198 p.

36. MORRIS, Kief. *Infrastructure as Code*. 2a ed. Estados Unidos: O'Reilly, 2020. 430 p.

37. NIXON, Robin. *Ubuntu: up and running*. 1a ed. Estados Unidos: O'Reilly, 2010. 464 p.

38. OSI. *The Open Source Definition.* [en línea]. <https://opensource.org/docs/osd>. [Consulta: 6 de diciembre de 2020].

39. PRITHVIRAJ, Sankaran. *Architecting Cloud SaaS software*. 1a ed. India: Pearson India, 2016. 216 p.

40. Project Management Institute. *A guide to the project management body of knowledge*. 6a ed. Estados Unidos: Project Management Institute, 2017. 760 p.

41. REICHLE, Meike. *Debian to adopt time-based releases.* [en línea]. <https://lwn.net/Articles/344007/>. [Consulta: 11 de enero de 2021].

42. SADOWSKI, Caitlin y ZIMMERMANN, Thomas. *Rethinking Productivity in Software Engineering*. 1a ed. Estados Unidos: Apress, 2019. 310 p.

43. SAMARA, Tarek. *ERP and Information Systems*. 1a ed. Reino Unido: John Wiley & Sons, 2015. 142 p.

44. SLASHDOT MEDIA. *Slashdot news.* [en línea]. <https://slashdot.org>. [Consulta: 4 de abril de 2021].

45. SPI. *Comité técnico de Debian.* [en línea]. <https://www.debian.org/devel/tech-ctte.es.html>. [Consulta: 4 de diciembre de 2020].

46. _____. *Constitución de Debian.* [en línea]. <https://www.debian.org/devel/constitution.es.html>. [Consulta: 29 de noviembre de 2020].

47. _____. *Debian.* [en línea]. <https://www.spi-inc.org/projects/debian/>. [Consulta: 4 de diciembre de 2020].

48. _____. *Debian - Donaciones.* [en línea]. <https://www.debian.org/donations.es.html>. [Consulta: 4 de diciembre de 2020].

49. _____. *Debian 10 Buster publicado.* [en línea]. <https://www.debian.org/News/2019/20190706.es.html>. [Consulta: 5 de diciembre de 2020].

50. _____. *Debian autobuilder network.* [en línea]. <https://www.debian.org/devel/buildd/>. [Consulta: 13 de marzo de 2021].

51. _____. *Debian mailing list for debian-user.* [en línea]. <https://lists.debian.org/debian-user/>. [Consulta: 4 de abril de 2021].

52. _____. *Debian mailing list for derivatives.* [en línea]. <https://lists.debian.org/debian-derivatives/>. [Consulta: 4 de abril de 2021].

53. _____. *Debian mirrors.* [en línea]. <https://www.debian.org/mirror/sponsors>. [Consulta: 14 de marzo de 2021].

54. _____. *Debian Policy Manual.* [en línea]. <https://www.debian.org/doc/debian-policy/index.html>. [Consulta: 7 de diciembre de 2020].

55. _____. *Debian Releases.* [en línea]. <https://www.debian.org/releases/>. [Consulta: 20 de diciembre de 2020].

56. _____. *Debian "testing" distribution.* [en línea]. <https://www.debian.org/devel/testing>. [Consulta: 21 de marzo de 2021].

57. _____. *Lintian Reports.* [en línea]. <https://lintian.debian.org/>. [Consulta: 4 de abril de 2021].

58. _____. *Organigrama de Debian.* [en línea]. <https://www.debian.org/intro/organization.es.html>. [Consulta: 29 de noviembre de 2020].

59. _____. *Procedimiento habitual de resoluciones.* [en línea]. <https://www.debian.org/vote/howto_follow.es.html>. [Consulta: 4 de diciembre de 2020].

60. _____. *¿Quién usa Debian?* [en línea]. <https://www.debian.org/users/index.es.html>. [Consulta: 29 de noviembre de 2020].

61. _____. *Setting up a Debian archive mirror.* [en línea]. <https://www.debian.org/mirror/ftpmirror>. [Consulta: 14 de marzo de 2021].

62. _____. *Status Debian Developer.* [en línea]. <https://nm.debian.org/public/people/dd_all/>. [Consulta: 29 de noviembre de 2020].

63. _____. *Status of the l10n in Debian.* [en línea]. <https://www.debian.org/international/l10n/po/rank>. [Consulta: 30 de noviembre de 2020].

64. _____. *The Debian Free Software Guidelines.* [en línea]. <https://www.debian.org/social_contract.en.html#guidelines>. [Consulta: 7 de diciembre de 2020].

65. _____. *Ultimate Debian Database.* [en línea]. <https://udd.debian.org/>. [Consulta: 13 de diciembre de 2020].

66. SRIVASTAVA, Manoj. *General Resolution: Editorial amendments to the social contract.* [en línea]. <https://www.debian.org/vote/2004/vote_003>. [Consulta: 6 de diciembre de 2020].

67. STARNES, Thomas. *Design Philosophy Behind Motorola's MC68000*. *Byte*. 8a ed. Estados Unidos: McGraw Hill, 1983. 656 p.

68. STROZZI, Francesco et al. *Scalable Workflows and Reproducible Data Analysis for Genomics*. *Evolutionary Genomics*. 2a ed. Estados Unidos: Springer New York, 2019. 780 p.

69. TEIXEIRA, José Apolinário y KARSTEN, Helena. *Managing to release early, often and on time in the OpenStack software ecosystem*. *Journal of Internet Services and Applications*. 10a ed. Reino Unido: Springer Open, 2019. 22 p.

70. THE CLOUD MARKET. *EC2 Statistics.* [en línea]. <https://thecloudmarket.com/stats#/by_platform_definition>. [Consulta: 4 de enero de 2021].

71. THE LINUX FOUNDATION. *Linux Foundation Training Prepares the International Space Station for Linux Migration.* [en línea]. <https://training.linuxfoundation.org/solutions/corporate-solutions/success-stories/linux-foundation-training-prepares-the-international-space-station-for-linux-migration/>. [Consulta: 4 de enero de 2021].

72. The StationerY Office. *ITIL Foundation*. 4a ed. Reino Unido: TSO Publishing, 2019. 212 p.

73. THYKIER, Niels. *Status on the stretch release.* [en línea]. <https://lists.debian.org/debian-devel-announce/2017/04/msg 00008.html>. [Consulta: 14 de marzo de 2021].

74. _____. *What is britney and britney2.* [en línea]. <https://release.debian.org/doc/britney/what-is-britney.html>. [Consulta: 4 de abril de 2021].

75. TOWNS, Anthony. *Social Contract GR's Affect on sarge.* [en línea]. <https://lists.debian.org/debian-vote/2004/04/msg00074.html>. [Consulta: 6 de diciembre de 2020].

76. VADAPALLI, Sricharan. *DevOps: Continuous Delivery, Integration, and Deployment with DevOps*. 1a ed. Reino Unido: Packt Publishing, 2018. 134 p.

77. WIRZENIUS, Lars. *About piuparts.debian.org.* [en línea]. <https://piuparts.debian.org/>. [Consulta: 23 de marzo de 2021].

APÉNDICES

Apéndice 1. **Plantilla de prueba de aceptación de usuario**

Nombre:	
Descripción:	
Estado inicial:	
Acciones por realizar:	
Resultado esperado:	

Ejemplo de prueba de aceptación considerando el formato anterior:

Nombre:	Inicio de sesión del usuario
Descripción:	El usuario puede iniciar sesión utilizando credenciales válidas
Estado inicial:	El usuario no ha iniciado sesión en el sistema. Las credenciales del usuario fueron previamente registradas
Acciones por realizar:	1. Encender la computadora 2. Ingresar el nombre de usuario 3. Ingresar la contraseña 4. Presionar la tecla de entrada
Resultado esperado:	El usuario correspondiente ha iniciado sesión en el sistema

Fuente: elaboración propia, empleando Microsoft Word.

Apéndice 2. **Plantilla de reporte de resultado de prueba de aceptación**

Nombre:	
Resultado:	
Comportamiento actual:	
Comentario:	

Ejemplo de reporte de resultado de prueba de aceptación considerando el formato anterior:

Nombre:	Inicio de sesión del usuario
Resultado:	Insatisfactorio
Comportamiento actual:	El sistema deja de responder al ingresar la contraseña del usuario y presionar la tecla de entrada
Comentario:	El problema ocurre para todos los usuarios normales pero no ocurre para el usuario administrador del sistema

Fuente: elaboración propia, empleando Microsoft Word.

Apéndice 3. **Porcentaje de actualizaciones**

Métrica:	Porcentaje de paquetes actualizados superior al 40 %
Fórmula de medición:	$\% \, de \, actualizaciones = \left(\frac{AM}{TA}\right)(100)$
Variable AM:	**A**ctualizaciones **M**ayores: cantidad de paquetes que han recibido una actualización mayor desde el inicio del ciclo de desarrollo (una actualización mayor consiste en una actualización que incluye nuevas funcionalidades)
Variable TA:	**T**otal **A**nterior: total de paquetes de software entregados en la versión anterior de Debian GNU/Linux
Inicio de la medición:	Inicio del ciclo de desarrollo
Frecuencia de verificación:	Mensual

Fuente: elaboración propia, empleando Microsoft Word.

Apéndice 4. **Porcentaje de paquetes nuevos**

Métrica:	Porcentaje de paquetes nuevos superior al 25 %
Fórmula de medición:	$\%\ \boldsymbol{de\ paquetes\ nuevos} = \left(\frac{\boldsymbol{N}}{\boldsymbol{TA}}\right)(\boldsymbol{100})$
Variable N:	**N**uevos: cantidad de paquetes de software nuevos que han sido agregados durante el presente ciclo de desarrollo
Variable TA:	**T**otal **A**nterior: total de paquetes de software entregados en la versión anterior de Debian GNU/Linux.
Inicio de la medición:	Inicio del ciclo de desarrollo
Frecuencia de verificación:	Mensual

Fuente: elaboración propia, empleando Microsoft Word.

Apéndice 5. **Producción mensual de paquetes nuevos**

Métrica:	Producción mensual de paquetes mayor o igual a 100 paquetes
Forma de medición:	Conteo de paquetes nuevos que alcanzaron la fase de producción durante el presente mes
Inicio de la medición:	Inicio del mes
Frecuencia de verificación:	Mensual

Fuente: elaboración propia, empleando Microsoft Word.

Apéndice 6. **Producción mensual de actualizaciones mayores**

Métrica:	Producción mensual de actualizaciones mayores igual o mayor que 100 paquetes
Forma de medición:	Conteo actualizaciones mayores que alcanzaron la fase de producción durante el presente mes
Inicio de la medición:	Inicio del mes
Frecuencia de verificación:	Mensual

Fuente: elaboración propia, empleando Microsoft Word.

Apéndice 7. **Cantidad mensual de errores reportados**

Métrica:	Cantidad mensual de errores reportados menor a 150
Forma de medición:	Conteo de errores de nivel serio o superior reportados durante el presente mes
Inicio de la medición:	Inicio del mes
Frecuencia de verificación:	Mensual

Fuente: elaboración propia, empleando Microsoft Word.

Apéndice 8. **Plantilla de listado de paquetes para entrega final**

Versión de Debian GNU/Linux:		
Fecha de creación del listado:		
Nombre clave:		
Nombre del paquete de software	**Versión del paquete de software**	**Responsable del paquete de software**

Ejemplo de listado de paquetes para entrega final considerando el formato anterior:

Versión de Debian GNU/Linux:	11	
Fecha de creación del listado:	2021-08-28	
Nombre clave:	Buster	
Nombre del paquete de software	**Versión del paquete de software**	**Responsable del paquete de software**
Bash	5.0-4	Matthias Klose
Grep	3.3-1	Anibal Monsalve
Gzip	1.9-3	Bdale Garbee
Hoteldruid	2.3.2-1	Marco De Santis
Perl	5.28.1-6	Niko Tyni

Fuente: elaboración propia, empleando Microsoft Word.