Universidad de las Ciencias Informáticas
Facultad 5

APLICACIÓN PARA AUTOMATIZAR LA CREACIÓN DE INSTALADORES DE LOS PRODUCTOS DEL PROYECTO DISEM EN LINUX

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Arturo Antonio Aguilera Jardines

Tutor: Ing. Alina Rodríguez Peña

La Habana, junio de 2016
"Año 58 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmo la presente a los ____ días del mes de _____ del año 2016.

Arturo Antonio Aguilera Jardines	Ing. Alina D. Rodríguez Peña
Autor	Tutor

DATOS DE CONTACTO

Tutor: Ing. Alina Dolores Rodríguez Peña.

Edad: 25.

Ciudadanía: Cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

Categoría Docente: Instructor.

E-mail: alina@uci.cu

Graduado de la UCI. Posee 5 años de experiencia como Analista de Sistema en proyectos de software de Informática Industrial y Realidad Virtual. Analista principal del proyecto DISEM (Diseño y Simulación de Estructuras Mecánicas).

DEDICATORIA

Dedico el fruto de mis horas de estudio a mis padres por ser tan geniales conmigo y con mi hermana. A mis abuelos por su constante preocupación. A mi tio Jorge y a mi tia Maria por ser especial conmigo y ayudarme en todo momento. A mi novia Yudith por ser tan especial en mi vida.
De Arturo.

AGRADECIMIENTOS

Llega el difícil momento de escribir los agradecimientos, llevar casi un año pensando en lo que quisiera poner no me ha servido de nada. Son tantos los que de una forma u otra han contribuido en esta tesis y en mi formación como profesional y ser humano que creo injusto el tener que mencionarlos, ya que siempre alguno, no por ser menos importante, quedará olvidado.

Primero que todo quiero agradecer a mi madre y a mi padre, que, a pesar de todos los obstáculos y dificultades por la cual se transitan por esta vida, siempre ha luchado por mí y por mi hermana entregando todo cuanto estuvo a su alcance, haciendo lo posible y lo imposible para apoyarnos siempre, con ese cariño infinito y esa confianza absoluta que siempre me han entregado.

Al amor de mi vida, a Yudith (mi feita, como cariñosamente me refiero a ella) la persona que siempre ha sabido apoyarme de todas las formas posibles. Cada vez que el mundo se me viene encima ahí está ella cargando una parte importante del peso que me toca.

Gracias por el amor, la confianza, la paciencia. Gracias por hacer mi vida más linda cada día. Eres muy especial, te adoro.

A mi tía María y a mi tío Jorge, que son las personas que han estado más cerca de mí, desde que comencé mis estudios universitarios, gracias por su apoyo, sin ustedes prácticamente este resultado no hubiese sido posible, los quiero mucho, para mí siempre serán mis segundos padres.

A todos mis familiares que de una forma u otra aportaron su granito de arena, en cada etapa de mi vida, a todos mis primos, mis tíos, mis abuelos, los quiero mucho.

A tos mis compañeras de estudios y a mis amigos siempre los tengo presente.

A la UCI, sin duda la mejor universidad del mundo, por darme la oportunidad de conocer a tanta gente linda, por todos los sueños hechos realidad, por los buenos y malos momentos, por los recuerdos que me llevo.

A mi tutora, Alina, mil gracias por todo, la paciencia, los conocimientos y su amistad. Sin tu ayuda nada de esto sería posible.

RESUMEN

El proceso de creación de paquetes de instalación en el proyecto DISEM se realiza de forma manual en un terminal a través de comandos del sistema operativo GNU/Linux, apoyándose en la herramienta *debhelper*, por alguno de los desarrolladores. Este proceso se torna engorroso por las configuraciones que se deben realizar, dependiendo en gran medida de la preparación con que cuente la persona encargada de realizar los instaladores para el proyecto. Para resolver el problema antes mencionado, la presente investigación tiene como objetivo desarrollar una aplicación de escritorio que automatice el proceso de creación de instaladores del proyecto DISEM en GNU/Linux. Se realizó un análisis a las soluciones similares para reconocer la existencia de herramientas diseñadas para la construcción de paquetes de instaladores. Se selecciona *dpkg* para la propuesta de solución. El desarrollo de la herramienta estuvo guiado por la metodología AUP – Variante UCI. Se realizaron pruebas funcionales y de aceptación, mediante la aplicación de la técnica de diseño de caja negra y el método de partición de equivalencia. La aplicación desarrollada permite, de forma visual, crear paquetes instaladores donde se gestionan las dependencias de bibliotecas externas para facilitar su despliegue, así como guardar o cargar la configuración de un proyecto lo que disminuye el tiempo de configuración de productos similares. De forma general se disminuyó la curva de aprendizaje por la complejidad de generar los instaladores para la plataforma GNU/Linux.

Palabras Clave: creación de paquetes, *dpkg*, instaladores, paquetes

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Cuba no está ajena de los avances tecnológicos del mundo, aunque no pueda acceder a las bondades de estos, debido al elevado costo de sus licencias. Por lo que en el país se han dedicado a desarrollar avances tecnológicos en diferentes ramas de la sociedad cubana. Un ejemplo de ello es el Asistente Mecánico (AsiXMec) en su versión 1.0 del proyecto de Diseño y Simulación de Estructuras Mecánicas (DISEM), desarrollado en el Centro de Entornos Interactivos 3D (Vertex) de la Universidad de las Ciencias Informáticas (UCI).

Actualmente en el proyecto DISEM se trabaja en el desarrollo de la versión AsiXMec 2.0, donde se incorporará el módulo de planos técnicos. El equipo de desarrollo se ha percatado desde la primera versión del software, que en el proceso de creación de AsiXMec es necesario la colaboración de especialistas CAD. Los usuarios finales de este tipo de aplicación son generalmente los ingenieros mecánicos. En la actualidad el proyecto no cuenta con ningún especialista en este campo para que valide la efectividad de sus productos, por lo que se ha tomado como alternativa que diseñadores del Ministerio de Industrias (MINDUS) puedan probar constantemente las mejoras del software. Para que los analistas y probadores miembros del equipo de desarrollo y posteriormente los diseñadores del MINDUS, tengan la versión más reciente en su puesto de trabajo sin necesidad de que cuenten con el código fuente de los productos, es necesario la creación de paquetes de instalación.

El proceso de creación de paquetes de instalación en el proyecto DISEM es realizado de forma manual (apoyándose en la herramienta *debhelper*), por uno de los desarrolladores, a través de comandos del sistema operativo GNU/Linux (actualmente AsiXMec solo está compilado para este sistema). Para construir un paquete de instalación Debian se deben configurar un conjunto de ficheros en el directorio “debian” que se crea en la raíz del proyecto. Estos ficheros contienen la información de las bibliotecas que necesita el software que se empaqueta, las direcciones hacia donde serán enviadas cuando se instalen, entre otros elementos. Es necesario además copiar las bibliotecas externas e internas del software para el directorio raíz donde se encuentre el código fuente del proyecto.

Este proceso se torna engorroso por las configuraciones que se deben realizar durante la ejecución del mismo. Para ello, es necesario la preparación con que cuente el desarrollador encargado de realizar los instaladores para el proyecto, del funcionamiento de la herramienta *debhelper* y los comandos necesarios para la creación de los mismos. En caso de ausencia de esta persona la creación de los instaladores se ve afectada, e influye en que no se tenga la última versión de los productos si se necesita hacerle alguna prueba o instalarlo en una máquina que no sea la del personal del proyecto (ferias, exposiciones de productos del centro, entre otros). Esto hace necesario tener que capacitar a otra persona para que asuma la tarea, lo que provoca tener que dedicar un esfuerzo extra. Si se desea realizar un paquete de instalación para otro producto o proyecto es necesario repetir todo el proceso de configuración de forma manual.

De ahí que surja la siguiente interrogante, que constituye el problema científico del presente trabajo: ¿Cómo automatizar el proceso de creación de instaladores del proyecto DISEM en GNU/Linux? Lo cual precisa como objeto de estudio: los procesos de automatización de creación de instaladores en GNU/Linux. El objetivo general que propone ésta investigación es: desarrollar una aplicación de escritorio que automatice el proceso de creación de instaladores del proyecto DISEM en GNU/Linux. Todo lo cual determina como campo de acción: los procesos de automatización de creación de herramientas de instaladores en GNU/Linux con interfaces visuales.

A continuación, se plantean un grupo de tareas que permitirán satisfacer durante el desarrollo de la investigación el objetivo planteado anteriormente:

- ✓ Elaboración del marco teórico de la investigación a partir del estado del arte existente sobre el tema actualmente.
- ✓ Selección de la herramienta de GNU/Linux que se utilizará para empaquetar el instalador del productor del proyecto.
- ✓ Determinación de los ficheros y configuraciones que necesita la herramienta para empaquetar.
- ✓ Descripción de los requisitos funcionales y no funcionales.
- ✓ Implementar la aplicación a través de los requisitos definidos.
- ✓ Ejecución de las pruebas y análisis de los resultados.

- ✓ Para la realización de la investigación y elaboración del presente trabajo se emplearon varios métodos científicos de investigación, entre los cuales se pueden mencionar los siguientes:

Métodos Teóricos

Histórico – Lógico: mediante este método científico se realizó un estudio del estado del arte, sobre los instaladores que se utilizan en la actualidad prestando especial atención a los instaladores de Windows y GNU/Linux. Permitió analizar la evolución de estos sistemas y cuál es su tendencia actual en el mundo.

Analítico – Sintético: el uso del método científico analítico–sintético permitió realizar un estudio por separado de cada una de los instaladores que más se utilizan en la actualidad, se definió que particularidades presentaban en común y se estableció una series de parámetros, atendiendo principalmente a las características relacionadas con sus objetivos fundamentales, para establecer una comparación entre ellas y tomar los resultados arrojados por dicha comparación, como datos de gran interés para la actual investigación.

Inducción – Deducción: mediante la aplicación del mismo se desarrolló un estudio con los principales instaladores existentes para sistemas operativos Windows y GNU/Linux, revisando sus características propias, basado en estas características se definieron características o cualidades que debe tener o cumplir el sistema que se propone en el presente trabajo.

Método de la modelación: para crear abstracciones con el objetivo de explicar la realidad, se utilizará para la modelación de los diversos diagramas necesarios en cada uno de los flujos de trabajo según la metodología seleccionada.

Métodos Empíricos

Consulta de la información: para realizar consultas de información en bibliográficas confiables, lo cual permite la elaboración del marco teórico de la investigación, y su orientación metodológica.

Observación: mediante el método científico de la observación se detectó la situación actual existente en el proyecto DISEM, en cuanto a las dificultades existentes a la hora de llevar a cabo el proceso de creación de paquetes de instalación, así como la necesidad de creación de una herramienta para cumplir con este requerimiento de forma automática.

Los posibles resultados al término de esta investigación son: aplicación de escritorio que automatice el proceso de creación de instaladores del proyecto DISEM en GNU/Linux, así como toda la documentación que generará el proceso investigativo, la cual servirá de material de consulta para futuras soluciones y la validación de resultados de la aplicación propuesta.

A continuación, se muestra la estructura del presente trabajo, incluyendo una síntesis de los capítulos y secciones fundamentales:

Capítulo 1. Fundamentación teórica.
En este capítulo se realiza un estudio acerca de las herramientas que efectúan la creación de instaladores, con la bibliografía consultada. Se analizan las ventajas y desventajas de cada una de ellas y si alguna cumple con las especificaciones que se necesitan en el centro Vertex para la creación de instaladores del proyecto de Diseño y Simulación de Estructuras Mecánicas (DISEM). Se exponen algunas de las principales características de estos instaladores, como funcionan y los diferentes tipos de ficheros instaladores que se conocen. Se aborda el tema sobre las metodologías de desarrollo, así como las herramientas y leguajes a utilizar.

Capítulo 2. Análisis y diseño de la solución.
Durante este capítulo se describe el sistema desde la perspectiva de Ingeniería de Software, usando el Proceso Unificado Ágil (AUP) como metodología. Se presenta el modelo de dominio del problema y se realiza la captura de requisitos y el modelo de casos de uso del sistema además de los artefactos generados en la fase de análisis y diseño propuestos por la metodología.

Capítulo 3. Implementación y validación de los resultados.

Se generan los productos de trabajo referentes a la fase de implementación y prueba del software. Como principales elementos se definen los diagramas de componentes y de despliegue. Se especifican los casos de pruebas aplicados a la solución desarrollada para validar su correcto funcionamiento.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordarán conceptos referentes al estudio del arte, además se realiza un análisis acerca de los distintos tipos de herramientas existentes para crear instaladores en el mundo, lo cual permitirá definir las ventajas y desventajas de cada uno de ellos y si alguno cumple con las especificaciones que se necesitan en el centro Vertex de la UCI, para la creación de instaladores del proyecto de DISEM. Se exponen algunas de las principales características de estos instaladores, como funcionan y los diferentes tipos de ficheros instaladores que se conocen. Se aborda el tema sobre las metodologías de desarrollo, así como las herramientas y leguajes a utilizar.

1.1. Instalador

Muchas veces un programa está formado por un conjunto de archivos. El programa en sí mismo solamente es uno, pero necesita de esos otros para funcionar. En muchas ocasiones, esos archivos tienen que estar colocados en diferentes partes del sistema, a veces relacionados con otros archivos. Además de esto, los programas tienen que registrarse en el registro de Sistema Operativo para que el sistema funcione correctamente. Todas estas tareas se pueden hacer manualmente, pero en muchas ocasiones resultan complejas y engorrosas. Es por eso que generalmente se utiliza un instalador: un programa especial que realiza todas esas tareas de manera automática (1).

La palabra instalar en el diccionario de la Real Academia de la Lengua Española se define como transferir al disco duro de una computadora un programa y prepararlo para su correcto funcionamiento (2).

En el libro *Beyond Software Architecture* del autor Luke Hohmann, definen un instalador como la herramienta que le permita al usuario final dar uso del software adicionándolo a la lista de programas que presente el sistema operativo en el cual trabaje (3).

En la presente investigación se utilizará la primera definición del concepto.

1.2. Empaquetado

El empaquetado de aplicaciones consiste en proporcionar las aplicaciones en forma de paquetes, a los que se suele llamar en inglés software *bundle* o *application bundle*. Estos paquetes están formados por los

programas ejecutables de la aplicación, así como por todas las bibliotecas de las que depende y otros tipos de ficheros (como imágenes, ficheros de audio, traducciones y localizaciones, etc.), de forma que se proporcionan como un conjunto. Las bibliotecas de las que depende el programa pueden haber sido enlazadas tanto de forma dinámica como también estática. Por tanto, el usuario percibe que el paquete como un conjunto que representa al programa en sí, cuando en realidad incluye varios ficheros (4).

1.3. Tipos de ficheros instaladores

Existen distintos tipos de ficheros de acuerdo al sistema operativo que se utilice, entre los principales se encuentran: bash, exe, .deb. A continuación, se describen cada uno de ellos.

BASH: Es un programa informático cuya función consiste en interpretar órdenes. Está basado en la *Shell* de Unix y es compatible con POSIX. Fue escrito para el proyecto GNU y es el intérprete de comandos por defecto en la mayoría de las distribuciones de GNU/Linux (5).

EXE: Los ficheros ejecutables o ficheros .exe contiene un programa que se ejecuta de manera automática al hacer doble clic sobre su icono (en Windows). Se utilizan en las plataformas MS-DOS y Windows (6).

.DEB: Es la extensión del formato de paquetes de software de la distribución de GNU/Linux, Debian GNU/Linux y derivadas (ej. Ubuntu), y, el nombre más usado para dichos paquetes. El programa predeterminado para manejar estos paquetes es *dpkg*, generalmente usando apt/aptitude, aunque también hay interfaces gráficas como *Synaptic, PackageKit, Gdebi* o en Ubuntu *Software Center* (7).

1.4. Programas generadores de instaladores

En cada plataforma existen herramientas o programas que tienen como función principal crear instaladores de programas. A continuación, se describen las que por sus características resultan de interés para la presente investigación.

InstallShield: Es una herramienta de software para la creación de instaladores o paquetes de software para Windows. También se puede emplear para administrar aplicaciones y paquetes de software en una amplia gama de dispositivos móviles y portátiles. *InstallShield* posee herramientas automatizadas para producir, empaquetar e instalar sus productos en formato MSI tradicional, se distribuye bajo licencia del tipo *node-lockede* incluye: panel para proyectos en Visual Basic 5.0 o 6.0, soporte para 29 idiomas y editor de sintaxis resaltado (8).

Set Up Factory: Es un excelente programa para la creación de archivos de instalación que no sólo compila y añade todos los ficheros necesarios para instalar el programa, sino que incluye avanzadas opciones de personalización. Esta herramienta posee una fácil interfaz capaz de crear instalaciones profesionales en poco tiempo, sin tener que complicarla con códigos de programación, provee instalaciones de un solo fichero así como las posibilidades de compresión, desinstalación, pantallas configurables, protección por contraseña, chequeo de caducidad de la instalación, soporte de múltiples idiomas; se distribuye bajo licencia Shareware (9).

Install Anywhere: Es un programa basado en Java disponible en dos ediciones (*Enterprise* y *Standard*). Proporciona a los profesionales un medio rápido y fácil de obtener instalaciones para software de las plataformas Solaris, HP-UX, AIX, GNU/Linux, i5/OS, Mac OS X, Windows entre otros. Es distribuido bajo licencia del tipo *node-locked* (de nodos bloqueados), la cual se distribuye a usuarios, sistemas operativos y ordenadores específicos (10).

Debhelper: Es un conjunto de herramientas que se utilizan para ayudar a construir un paquete Debian. La filosofía detrás de *Debhelper* es proporcionar una colección de pequeña, simple, y fácil de entender las herramientas que se usan en Debian / rules para automatizar diversos aspectos comunes de la construcción de un paquete (11).

Dpkg: Es un equipamiento lógico utilizado para la gestión de paquetería en Debian GNU/Linux y distribuciones de GNU/Linux, como es el caso de Ubuntu GNU/Linux. Es una herramienta de bajo nivel que permite la instalación, desinstalación y consulta de información de paquetes con extensión *.deb*. Herramientas de alto nivel como *apt, aptitude o synaptic* lo utilizan, siendo está dos últimas las más sofisticadas. *dpkg* es una herramienta de nivel medio para instalar, construir, borrar y gestionar los paquetes de Debian GNU/Linux (12).

Fakeroot: *fakeroot* es una herramienta de Debian que ejecuta una orden en un entorno donde parece que se tiene permisos de superusuario para la manipulación de ficheros. Útil para permitir a usuarios crear archivos (*tar, ar, .deb* etc.) con ficheros con permisos/propietarios de superusuario (13).

1.5. Selección de la herramienta para la creación del instalador

Dadas las herramientas anteriormente descritas se hizo necesario establecer una comparación de acuerdo a las principales características que requiere el equipo de desarrollo. Las plataformas de instalación y desarrollo, el soporte a diversos lenguajes de programación de las aplicaciones a instalar y la licencia de distribución fueron los principales criterios a medir. En la Tabla 1 que se muestra a continuación se puede apreciar una comparación teniendo en cuenta si las herramientas analizadas están disponibles en las plataformas en las que está interesado el equipo de desarrollo.

Tabla 1. Comparación de las herramientas.

Producto	Plataformas en las que está disponible		Soporte a diversos lenguajes	Tipo de licencia	Instalador gráfico	Dependencias
	Windows	GNU/Linux				
InstallShield 2012 Spring	✓		✓	Privada (node-lockede)	✓	
Set Up Factory 8.2.2.0	✓			Privada (Shareware)	✓	
Install Anywhere 7	✓	✓		Privada (node-lockede)	✓	
dpkg		✓	✓	GPL-2+ BSD-2-clause (Incluyendo otras)		libbz2-1.0 libc6 liblzma5 libselinux1 zlib1g tar
Debhelper		✓		GPL-3.0+ GPL-2.0+ (Incluyendo otras)		dpkg-dev binutils po-debconf man-db libdpkg-perl utotools-dev dh-autoreconf
fakeroot		✓	✓	Copyright 1997, 1998, 1999, (Incluyendo otras)		libc6 libfakeroot

Las características evaluadas en la tabla anterior, conllevan a que se generen instaladores en cada uno de estos sistemas operativos. Según la información brindada se dejan de analizar las *Install Shield* 2012 *Sprind* y *Set Up Factory* 9, ya que teniendo en cuenta las políticas de migración al software libre del

proyecto el entorno de desarrollo debe ser en el sistema operativo GNU/Linux, y estas son herramientas que solo se pueden encontrar disponibles en los sistemas operativos de la corporación Microsoft. Se elimina del proceso de selección la herramienta *Install Anywhere* 7, ya que es una herramienta que se distribuye bajo licencias de tipo privativas y adquirirlas sería un monto adicional al costo del proyecto. Por lo que se ha seleccionado la herramienta *dpkg*, cumpliendo la misma con las características y requisitos necesarios para llevar a cabo el desarrollo de dicho proyecto.

1.5.1. Especificaciones de la herramienta seleccionada.

NOMBRE: *dpkg-deb* es una herramienta de manipulación de paquetes Debian *(*archivos **.deb).*
SINOPSIS: *dpkg-deb* [opción ...]
Comandos utilizados: -b, directorio --build [archivo | *directorio*]: crea un archivo de Debian desde el árbol de archivos almacenados en el directorio. El *directorio* debe tener un subdirectorio DEBIAN, que contiene los archivos de información de control, como el archivo de *control* de sí mismo. Este directorio no aparecerá en el paquete binario generado, pero en cambio, los archivos contenidos en él serán puestos en el área de información de control del paquete binario, a menos que se especifique --nocheck, dpkg-deb leerá DEBIAN/control y lo analizará.

1.6. Herramientas de desarrollo.

1.6.1. Qt Creator

Qt Creator ha sido desarrollado para ser un entorno de desarrollo integrado (IDE) multiplataforma adaptado a las necesidades de los desarrolladores de Qt. Qt Creator se ejecuta en los sistemas operativos de escritorio Windows, GNU/Linux/X11 y Mac OS X y permite a los desarrolladores crear aplicaciones para múltiples escritorios y plataformas de dispositivos móviles. Algunas de las características de *Qt Creator* son (14):

- ✓ El editor de código avanzado de *Qt Creator* ofrece compatibilidad con la edición del lenguaje C++ y QML (JavaScript), ayuda sensible al contexto y finalización de código.
- ✓ Ofrece dos editores visuales integrados, Qt Designer para la creación de interfaces de usuario de widgets Qt y *Qt Quick Designer* para el desarrollo de interfaces de usuario animadas con el lenguaje QML.
- ✓ Permite la administración de proyectos y versiones.

- ✓ Ofrece compatibilidad con la creación y ejecución de aplicaciones Qt para ordenadores de escritorio y dispositivos móviles. La configuración generada te permite cambiar rápidamente entre los destinos de generación.

Se selecciona esta herramienta por las características antes mencionadas y por ser además la que se utiliza en el proyecto DISEM.

1.7. Metodologías de Desarrollo

En el proceso de construcción de un software las metodologías de desarrollo de software constituyen un pilar fundamental evitando que este vaya rumbo al fracaso. Se definen como un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto, pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales. Se han desarrollado en los últimos años dos corrientes en lo referente a los procesos de desarrollo, los llamados métodos (o metodologías) pesados y los ligeros (o ágiles).

1.7.1. Metodologías pesadas o tradicionales

Las metodologías pesadas o tradicionales son más eficaces y necesarias cuanto mayor es el proyecto que se pretenda realizar respecto a tiempo y recursos que son necesarios emplear. Se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, ya que pretenden prever todo de antemano (15).

Dentro del grupo de metodologías que encierra esta clasificación la más utilizada a nivel mundial es el Proceso Unificado de Rational o RUP[1]. Las principales características de esta metodología para el desarrollo de software es realizar un proceso iterativo e incremental, centrado en la arquitectura y guiado por casos de uso. Permite llevar un control de cambios estricto y divide el proceso de desarrollo en cuatro fases por las cuales se transita varias veces: inicio, elaboración, desarrollo y transición (16).

1.7.2. Metodologías ágiles o ligeras

[1] RUP, acrónimo del inglés Rational Unified Process

Las metodologías ágiles se encargan de valorar al individuo y las iteraciones del equipo más que a las herramientas o los procesos utilizados, priorizan la creación del software ante la generación excesiva de documentación, el cliente está en todo momento colaborando en el proyecto y la capacidad de respuesta ante un cambio realizado es más importante que el seguimiento estricto de un plan (15).

Dentro de las metodologías ágiles se encuentra el Proceso Unificado Ágil o AUP[2] la cuál es una versión simplificada de RUP. AUP dispone de cuatro fases de trabajo al igual que RUP: inicio, elaboración, construcción y transición. Abarca siete flujos de trabajo, cuatro ingenieriles y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente respectivamente. El Modelado agrupa los tres primeros flujos de RUP (Modelado del negocio, Requerimientos, Análisis y Diseño).

1.7.3. Variación AUP – UCI

En la UCI se realizó una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Las fases que propone son:

- ✓ Inicio: Se llevan a cabo las actividades relacionadas con la planificación del proyecto.
- ✓ Ejecución: Se ejecutan las actividades requeridas para desarrollar el software (incluye entre otras actividades el ajuste de los planes del proyecto, el modelado del negocio, obtención de requisitos, implementación y liberación el producto.
- ✓ Cierre: Se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Las disciplinas que propone AUP-UCI son: Modelado de negocio, Requisitos, Análisis y diseño Implementación, Pruebas Internas, de Liberación y Aceptación. La selección de la metodología a utilizar se hace en base a las características del equipo y las necesidades específicas de la situación. Para elegir una metodología de desarrollo de software se debe tener en cuenta dos factores fundamentales: el tipo de proyecto que se desea desarrollar y el tiempo que se dispone para desarrollar el mismo.

A partir del análisis realizado a las metodologías pesadas y ligeras y las necesidades del proyecto, se propone utilizar la metodología AUP en la variación de AUP-UCI, por ser la que más se adapta al proyecto

[2] AUP, acrónimo del inglés Agile Unified Process

a desarrollar y a las condiciones de trabajo. Además, se utilizará para guiar el ciclo de vida del software a desarrollar, ya que el producto forma parte de la Universidad.

1.8. Lenguaje de modelado y herramienta CASE

1.8.1. Lenguaje de modelado UML 2.0

UML es uno de los lenguajes más reconocidos y utilizados en la actualidad para la modelación de software. Se puede aplicar de varias formas para sobrellevar una metodología de desarrollo, aunque no define cuál usar o qué proceso emplear. Ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables (17). Contempla diversos tipos de diagramas entre los que se encuentran:

- ✓ Diagramas de clases para personalizar la estructura estática de las clases en el sistema.
- ✓ Diagramas de objetos para simbolizar la estructura estática de los objetos en el negocio.
- ✓ Diagramas de casos de uso para representar los procesos del negocio.
- ✓ Diagramas de actividad para modelar el comportamiento de los casos de uso, objetos u operaciones.
- ✓ Diagramas de colaboración para modelar interacciones entre objetos.
- ✓ Diagramas de componentes para modelar componentes.
- ✓ Diagramas de implementación para personificar la distribución del sistema.

1.8.2. Herramientas CASE

Las herramientas CASE[3] constituyen un conjunto de programas y ayudas que suministran apoyo a los ingenieros de software y desarrolladores. Estas herramientas tienen como objetivo principal computarizar y apoyar una o más fases del ciclo de vida del perfeccionamiento de sistemas, acelerando el proceso. Siendo además la aplicación de conocimientos informáticos a las actividades, las técnicas y las metodologías de desarrollo.

Fue seleccionada como herramienta CASE Visual Paradigm para UML, en su versión 8.0 la cual es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y

[3] CASE, acrónimo del inglés Computer Aided Software Engineering

diseño orientados a objetos, implementación, pruebas y despliegue. Contiene una amplia variedad de funcionalidades: realizar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar la documentación necesaria. Su diseño está centrado en los casos de uso y enfocado al negocio y permite la integración a los principales IDE.

1.9. Lenguajes de programación

Para la implementación de la herramienta se utilizan los lenguajes Bash y C++, ya que el primero es para el trabajo en líneas de comando e interacción con el sistema y el segundo para implementar las clases de la solución propuesta.

1.9.1. Bash

Es el shell o intérprete de lenguaje de comandos para el sistema operativo GNU/Linux. El nombre es un acrónimo de "*Bourne-Again SHell*". En la actualidad este se ejecuta en la mayoría de las versiones de Unix y en otros sistemas como MS-DOS y Windows.

Las principales características del intérprete BASH son:

- ✓ Ejecución síncrona de órdenes (una tras otra) o asíncrona (en paralelo).
- ✓ Distintos tipos de redirecciones de entradas y salidas para el control y filtrado de la información.
- ✓ Ejecución de mandatos interactiva y desatendida, aceptando entradas desde teclado o desde ficheros.
- ✓ Proporciona una serie de órdenes internas para la manipulación directa del intérprete y su entorno de operación.

Un lenguaje de programación de alto nivel, que incluye distintos tipos de variables, operadores, matrices, estructuras de control de flujo, entrecomillado, sustitución de valores y funciones.

1.9.2. C++

Es un lenguaje de programación, orientado a objetos derivado del C, en realidad un súper conjunto de C, que nació para añadirle cualidades y características de las que carecía. Las principales características de C++ son el soporte para programación orientada a objetos, el soporte de plantillas o programación genérica (*templates*), la portabilidad, brevedad, programación modular, compatibilidad con C y velocidad.

Abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Además, se trata de un lenguaje de programación estandarizado (ISO/IEC 14882:1998), ampliamente difundido, y con una biblioteca estándar C++, que lo ha convertido en un lenguaje universal, de propósito general, y ampliamente utilizado tanto en el ámbito profesional como en el educativo (18).

1.10. Consideraciones parciales.

Con la realización del presente capítulo han quedado definidos los conceptos más importantes para el correcto entendimiento de los términos usados en la presente investigación. El análisis a las soluciones similares sirvió para reconocer la existencia de herramientas diseñadas para la construcción de instaladores y también permitió determinar las ventajas de su utilización en el proceso de desarrollo. Se selecciona AUP – UCI como metodología de desarrollo, *dpkg* como herramienta para la construcción del instalador, UML, Bash y C++ como lenguaje de modelado y programación respectivamente y Visual Paradigm como herramienta CASE.

CAPÍTULO 2. ANÁLISIS Y DISEÑO DE LA SOLUCIÓN PROPUESTA

Durante este capítulo se describe el sistema desde la perspectiva de Ingeniería de Software, usando el Proceso Unificado Ágil (AUP) como metodología. De los cuatro escenarios que propone la metodología para modelar un sistema en los proyectos se utilizará el Escenario No 1 modelando con casos de usos del sistema. Se presenta el modelo de dominio del problema y se realiza la captura de requisitos y el modelo de casos de uso del sistema además de los artefactos generados en la fase de Análisis y Diseño propuestos por la metodología.

2.1. Modelo conceptual

El modelo del conceptual es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real en lugar de componentes de software. A continuación, se muestra el modelo conceptual de la aplicación propuesta (ver Figura 1). Este se basa en obtener una aplicación capaz de crear paquetes de instalación de productos. Al crear un paquete se tienen configuraciones necesarias para su ejecución, pero además se pueden administrar configuraciones opcionales. Dentro de estas configuraciones se permite la gestión de bibliotecas y complementos adicionales, ejecutables auxiliares y dependencias con otras aplicaciones.

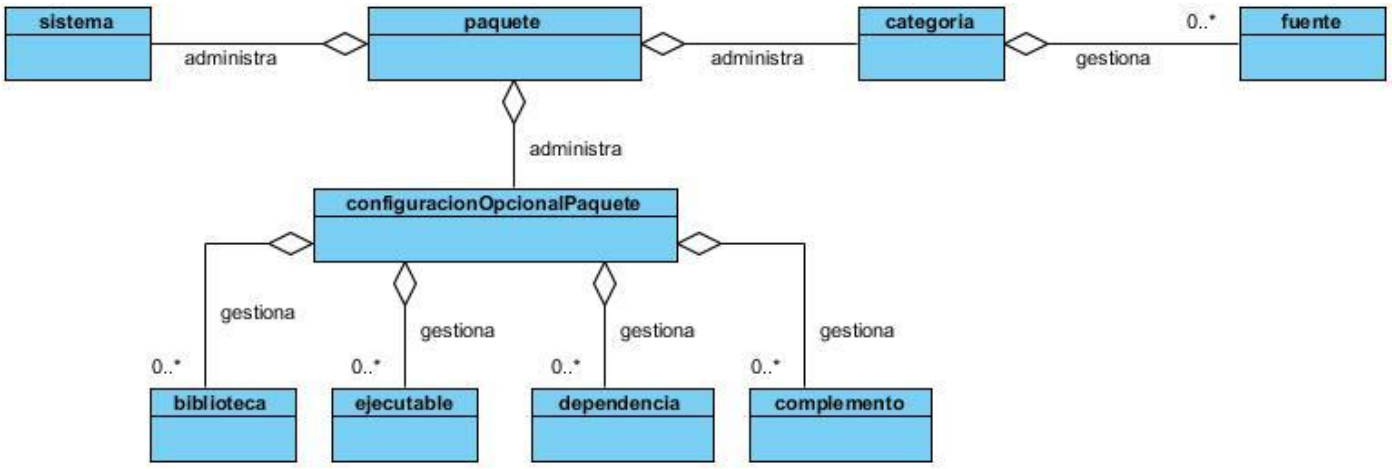

Figura 1. Modelo conceptual

2.2. Descripción de la propuesta de solución

La herramienta que se debe obtener como resultado de la investigación debe permitir la creación de paquetes de instalación teniendo en cuenta las configuraciones necesarias para su confección. Se deben brindar dos opciones para la creación del paquete, a partir del código fuente o a partir de un binario. La herramienta debe permitir que el usuario configure el nombre del paquete, la versión del software y una descripción del mismo, así como la dirección donde se creará el paquete que se generará. Se deben configurar además las dependencias del paquete, la categoría a la que pertenece (Gráficos, Internet, Aplicaciones, por ejemplo), la arquitectura del procesador (amd64 o i386), y otros elementos como el nombre e íconos del programa, la página oficial del software, los ejecutables auxiliares, las bibliotecas y complementos adicionales, entre otros.

2.3. Requisitos del sistema

Para lograr que el desarrollo de un software tenga éxito, es esencial comprender perfectamente los requisitos del software. La captura de los requisitos va a influenciar en todo el proceso de desarrollo del software, repercutiendo en el resto de las fases de desarrollo. Una definición eficiente de los requisitos funcionales y no funcionales, permitirá al equipo de desarrollo reducir los riesgos que se puedan presentar durante la implementación del sistema y se puede determinar lo que el sistema debe hacer. Además, de establecer aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes del mismo (19).

2.3.1. Requisitos Funcionales

Los requisitos funcionales especifican acciones que debe poder realizar un sistema, sin tener en cuenta las restricciones físicas, además definen su comportamiento de salida y entrada. Se mantienen invariables sin importar con que propiedades o cualidades se relacionen. A continuación se presentan los requerimientos funcionales tomados en cuenta durante el desarrollo del presente trabajo (ver Tabla 2).

Tabla 2. Requisitos funcionales

No	Nombre	Descripción	Complejidad
RF-1	Administrar configuración del paquete de instalación.	El sistema permite administrar la configuración necesaria para la creación de un paquete de instalación. Para ello se tiene en cuenta: el nombre, la versión y descripción del paquete, además de la ruta del ejecutable principal y la ruta del paquete generado.	Alta
RF-2	Administrar configuración opcional del paquete de instalación	El sistema permite administrar la configuración opcional para la creación de un paquete de instalación. Para ello se tiene en cuenta: las dependencias de los paquetes, las categorías de las aplicaciones, la arquitectura del procesador de la computadora, el mantenedor del paquete, la página oficial del software, nombre e íconos del programa, los ejecutables auxiliares, las bibliotecas y complementos adicionales, traducción de las aplicaciones, ícono de programas y script de post-instalación.	Alta
RF-3	Adicionar ejecutables auxiliares.	El sistema permite adicionar la ruta de los ejecutables auxiliares del software en la computadora. Estos puedes ser scripts, bash, Python, Perl, entre otros.	Media
RF-4	Modificar ejecutables auxiliares.	El sistema permite modificar la ruta de el/los binario(s) auxiliar(es) del software previamente seleccionado.	Media
RF-5	Eliminar ejecutables auxiliares.	El sistema permite eliminar la ruta de el/los binario(s) auxiliar(es) del software previamente seleccionado.	Media
RF-6	Adicionar dependencias.	El sistema permite adicionar una dependencia del software en la computadora. En este se introducen los nombres de las aplicaciones que son necesarias para crear el paquete de instalación.	Media

RF-7	Modificar dependencias.	El sistema permite modificar una dependencia del software previamente seleccionada.	Media
RF-8	Eliminar dependencias.	El sistema permite eliminar una dependencia del software previamente seleccionada.	Media
RF-9	Adicionar bibliotecas adicionales.	El sistema permite adicionar bibliotecas adicionales del software en la computadora.	Media
RF-10	Modificar bibliotecas adicionales.	El sistema permite modificar bibliotecas adicionales previamente seleccionadas.	Media
RF-11	Eliminar bibliotecas adicionales.	El sistema permite eliminar bibliotecas adicionales previamente seleccionadas.	Media
RF-12	Adicionar complementos adicionales.	El sistema permite adicionar complementos adicionales del software en la computadora.	Media
RF-13	Modificar complementos adicionales.	El sistema permite modificar complementos adicionales previamente seleccionadas.	Media
RF-14	Eliminar complementos adicionales.	El sistema permite eliminar complementos adicionales previamente seleccionadas.	Media
RF-15	Administrar opciones del sistema.	El sistema permite seleccionar las opciones si se desea que la aplicación sea gráfica o por consola, además si esta se ejecutará con permisos de administración.	Media

RF-16	Administrar categoría específica de proyecto	El sistema permite administrar la categoría específica del proyecto AsiXMec. Para ello se tiene en cuenta: las fuentes, la carpeta principal.	Media
RF-17	Adicionar fuentes.	El sistema permite adicionar fuentes del software en la computadora.	Media
RF-18	Modificar fuentes.	El sistema permite modificar fuentes previamente seleccionadas.	Media
RF-19	Eliminar fuentes.	El sistema permite fuentes adicionales previamente seleccionadas.	Media
RF-20	Salvar configuración del paquete de instalación.	El sistema permite salvar la configuración del paquete de instalación en la computadora.	Baja
RF-21	Cargar configuración del paquete de instalación.	El sistema permite cargar la configuración del paquete de instalación existente en la computadora.	Baja

2.3.2. Requisitos no Funcionales

Los requisitos no funcionales hacen relación a las características del sistema que aplican de manera general como un todo, más que a rasgos particulares del mismo. Un requisito no funcional especifica los criterios que se deben usar para juzgar el funcionamiento de un sistema y no su comportamiento específico (20). A continuación, se muestran los requisitos no funcionales de la propuesta de solución (ver tablas 3, 4, 5, 6, 7 y 8).

Tabla 3. RNF- Usabilidad/Aprendibilidad.

Atributo de Calidad	Usabilidad.
Sub-atributos/Sub-características	Aprendibilidad.
Objetivo	Garantizar que toda persona con conocimientos básicos pueda utilizar la aplicación.
Origen	Humano.
Artefacto	Aplicación.
Entorno	La aplicación está funcionando correctamente.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Realizar un flujo de trabajo en la aplicación.	
El usuario con conocimientos básico debe trabaja en la aplicación.	La aplicación realiza el flujo de trabajo del usuario.
Medida de respuesta	
Disponibilidad de la aplicación.	

Tabla 4. RNF- Usabilidad/Agradabilidad.

Atributo de Calidad	Usabilidad.
Sub-atributos/Sub-características	Agradabilidad.
Objetivo	Garantizar que la aplicación sea agradable para el usuario.
Origen	Humano.
Artefacto	Aplicación.
Entorno	La aplicación está funcionando correctamente.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Realizar un flujo de trabajo en la aplicación.	
Las interfaces de la aplicación deben	La aplicación es agradable y atractiva.

brindar al usuario una experiencia agradable y atractiva visualmente.	
Medida de respuesta	
Disponibilidad de la aplicación.	

Tabla 5. RNF- Portabilidad/Adaptabilidad.

Atributo de Calidad	Portabilidad.
Sub-atributos/Sub-características	Adaptabilidad.
Objetivo	Garantizar la ejecución de la aplicación en ambientes GNU/Linux y con hardware específico.
Origen	Humano.
Artefacto	Aplicación.
Entorno	La aplicación está funcionando correctamente.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Realizar un flujo de trabajo en la aplicación.	
La aplicación debe ejecutarse en diferentes distribuciones de GNU/Linux tales como: ✓ Debian 8. ✓ Nova 5.0. ✓ Ubuntu 14.04 o superior. La aplicación debe ejecutarse en hardware con: ✓ Procesador: Intel Core i3 a 3.3 GHz o superior. ✓ Memoria RAM: 2.0 GB Mínimo.	La aplicación funciona correctamente en el entorno.
Medida de respuesta	
Disponibilidad de la aplicación.	

Tabla 6. RNF- Portabilidad/Instalabilidad.

Atributo de Calidad	Portabilidad.
Sub-atributos/Sub-características	Instalabilidad.
Objetivo	Garantizar la instalación de la aplicación.
Origen	Humano.
Artefacto	Aplicación.
Entorno	La aplicación está fuera de línea.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Instalar aplicación.	
Para instalar la aplicación se debe tener instalado el módulo *QWidget* del framework *Qt*.	La aplicación se instala correctamente.
Medida de respuesta	
Disponibilidad de la aplicación.	

Tabla 7. RNF- Mantenibilidad/Comprobabilidad.

Atributo de Calidad	Mantenibilidad.
Sub-atributos/Sub-características	Comprobabilidad.
Objetivo	Garantizar que la aplicación funcione durante la ejecución de las pruebas funcionales.
Origen	Humano.
Artefacto	Aplicación.
Entorno	La aplicación está funcionando correctamente.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Pruebas funcionales.	
La aplicación debe permitir la ejecución de las pruebas funcionales para encontrar defectos.	La aplicación funciona correctamente antes las pruebas.
Medida de respuesta	

Disponibilidad de la aplicación.

Tabla 8. RNF- Eficiencia/Tiempo.

Atributo de Calidad	Eficiencia.
Sub-atributos/Sub-características	Tiempo.
Objetivo	Garantizar que la aplicación se instale en un tiempo mínimo. Garantizar que la aplicación realice el empaquetamiento en un tiempo mínimo.
Origen	Humano.
Artefacto	Aplicación.
Entorno	La aplicación está fuera de línea.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Instalar aplicación.	
La aplicación debe instalarse en el rango de tiempos 3 a 5 segundos.	La aplicación se instala en el rango de tiempo.
1. b Empaquetar.	
La aplicación debe instalarse en el rango de tiempos 3 a 5 minutos.	La aplicación realiza el empaquetado en el rango de tiempo.
Medida de respuesta	
Disponibilidad de la aplicación.	

2.4. Modelo de Casos de Uso

La forma en que los actores usan el sistema es representada a través de los casos de usos. Estos son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del actor. En los siguientes epígrafes se muestran los actores y los casos de uso identificados en el presente trabajo.

2.4.1. Actores del Sistema

Los actores son las personas, sistemas o hardware externo que interactuará con la aplicación (21). En este caso quien hará uso del sistema será un usuario genérico, en esta clasificación entrarán todas las personas que interactúen con la aplicación (ver Tabla 9) .

Tabla 9. Actores del sistema.

Actor	Descripción
Usuario	Es el encargado de interactuar con la aplicación, podrá acceder a todas las funcionalidades presentes en la misma.

2.4.2. Diagrama de Casos de Uso del Sistema

El Diagrama de Casos de Uso del Sistema (DCUS) representa gráficamente los casos de uso y su interacción con los actores (ver Figura 2). El usuario puede administrar la configuración del paquete de instalación a través de sus parámetros necesarios para su ejecución y la gestión de bibliotecas, complementos, ejecutables auxiliares y dependencias.

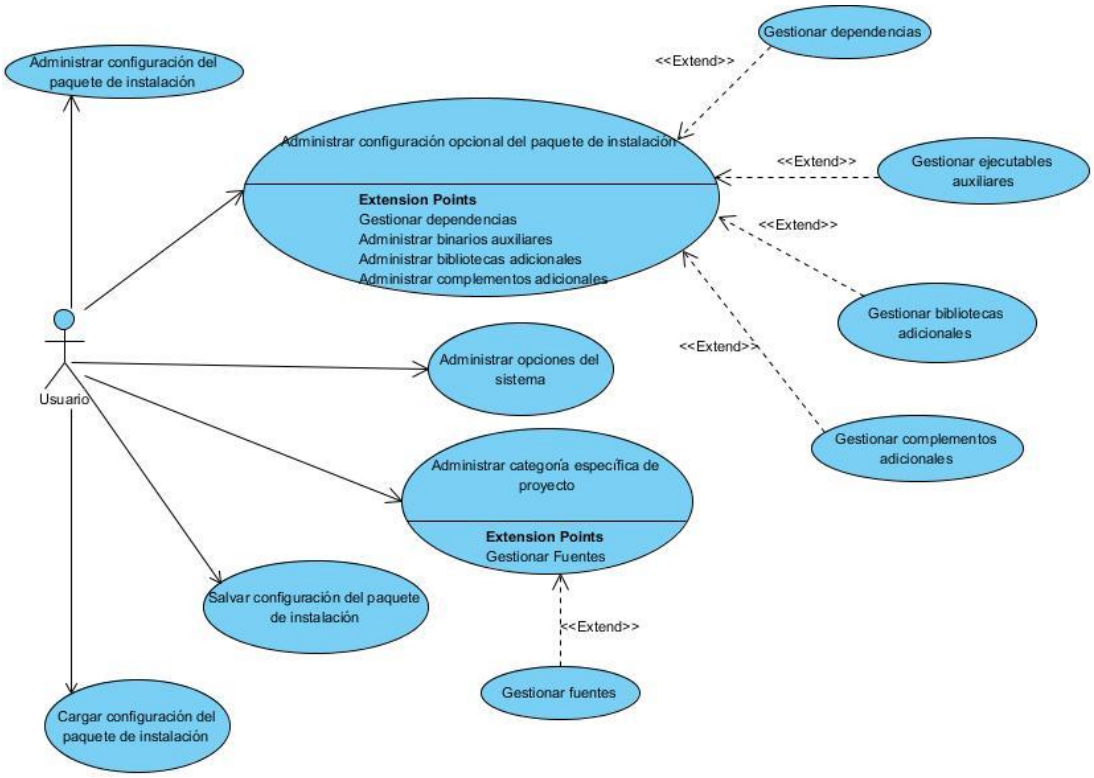

Figura 2. Diagrama de casos de uso.

2.4.3. Descripción de los Casos de Uso del Sistema

La descripción de los casos de uso permite comprender mejor el funcionamiento de un sistema. Muestran cómo debería reaccionar el mismo ante una entrada del usuario, así como la descripción de las funcionalidades con las que cuenta. A continuación se muestra la descripción de los casos de uso Administrar opciones del paquete de instalación y Configurar paquete de instalación (ver Tabla 10 y Tabla 11), el resto de las descripciones de los casos de uso se encuentra en el ANEXO 1: Descripción de casos de uso

Tabla 10. Administrar configuración opcional del paquete de instalación.

Caso de Uso	Administrar configuración opcional del paquete de instalación.	
Objetivo	Administrar la configuración opcional del paquete de instalación de productos.	
Actores	Usuario	
Resumen	El caso de uso se inicia cuando el usuario del sistema accede a la opción Administrar opciones del paquete de instalación. El sistema permite introducir los datos necesarios para la configuración del paquete de instalación, el usuario introduce los datos necesarios y el caso de uso finaliza.	
Complejidad	Baja	
Prioridad	Alta	
Precondiciones	Debe existir el software, el cual se realizará empaquetado.	
Poscondiciones	Se realizó la configuración opcional del paquete de instalación de productos.	
Flujo de eventos		
Flujo básico Administrar opciones		
	Actor	**Sistema**
1.	Selecciona la opción de “Opción”	
2.		Muestra los campos: ✓ Dependencia (ver CU Gestionar dependencias) ✓ Categoría. ✓ Arquitectura. ✓ Autor. ✓ Página oficial. ✓ Nombre del programa. ✓ Imágenes del programa. ✓ Binarios Auxiliares. (ver CU Gestionar binarios auxiliares). ✓ Iconos del programa. ✓ Bibliotecas adicionales. (ver CU Gestionar bibliotecas adicionales) ✓ Complementos adicionales (ver CU Gestionar complementos adicionales). ✓ Traducciones.

		✓ Post install script.
3.	Selecciona y llena los campos opcionales necesarios.	
4.		Finaliza el caso de uso.
Prototipo funcional		
Relaciones	**CU incluidos**	No aplica
	CU extendidos	-CU Gestionar dependencias. -CU Gestionar ejecutables auxiliares. -CU Gestionar bibliotecas adicionales. -CU Gestionar complementos adicionales.
Requisitos no funcionales	No aplica	
Asuntos pendientes	No aplica	

Tabla 11. Administrar configuración del paquete de instalación.

Caso de Uso	Administrar configuración del paquete de instalación.	
Objetivo	Administrar la configuración del paquete de instalación de productos.	
Actores	Usuario	
Resumen	El caso de uso se inicia cuando el usuario accede a la opción Configurar el paquete de instalación de la aplicación, el sistema permite introducir los datos necesarios para la configuración, el usuario introduce los datos necesarios y el caso de uso finaliza.	
Complejidad	Media	
Prioridad	Media	
Precondiciones	Para poder configurar el paquete de instalación debe existir el ejecutable del software en la computadora.	
Postcondiciones	Se realizó la configuración del paquete de instalación de productos.	
Flujo de eventos		
Flujo básico Configurar paquete de instalación		
	Actor	**Sistema**
1.	Accede a la opción “Configuración” .	
2.		Muestra los campos: ✓ Nombre del paquete. ✓ Versión del software. ✓ Descripción del software. ✓ Ejecutable principal del paquete ✓ Guardar en.
3.	Inserta los datos en los campos.	
4.		Finaliza el caso de uso.
Relaciones	**CU incluidos**	No aplica

	CU extendidos	No aplica
Requisitos no funcionales	No aplica	
Asuntos pendientes	No aplica	

2.5. Arquitectura de la solución

El diseño arquitectónico representa la estructura de los datos y componentes del programa necesarios para construir un sistema computacional. Según Pressman, la arquitectura es la forma en que se integran los diversos componentes del sistema para formar un todo cohesionado. Es la manera en que la solución se amolda y combina con aplicaciones externas. Resumiendo, la arquitectura de software de un programa es la estructura del sistema que incluyen los componentes de software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos. Permite a un ingeniero de software analizar la efectividad del diseño para cumplir con los requisitos establecidos, considerar opciones arquitectónicas en una etapa en la que aún resulta relativamente fácil hacer cambios al diseño y reducir los riesgos asociados a la construcción del software (21).

Para el desarrollo de la solución se adoptará una arquitectura en Capas, dada las potencialidades que ofrece para la reutilización y el aprovechamiento de los beneficios de una clara separación entre las funciones desplegadas en capas. Además, permite la estandarización y proporciona una distribución clara del trabajo entre los miembros de un equipo de desarrollo. El patrón arquitectónico a utilizar es el Modelo-Vista-Controlador.

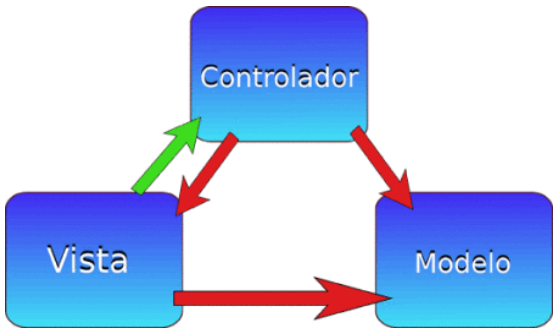

Figura 3. Arquitectura de la solución.

Vista: contiene las vistas de la aplicación, es la capa que interactúa con el usuario.

Controlador: contiene la clase controladora de la aplicación, esta interactúa con las capas Vista y Modelo.

Modelo: contiene las clases del negocio, y todas finalizan con *DataModel*.

Al definirse la arquitectura global de la aplicación se procede a la confección del diagrama de paquetes correspondiente a la misma, a partir de una explicación detallada de las relaciones existentes entre las clases que lo conforman.

2.5.1. Diagrama de Clases del Diseño

En el flujo de trabajo de Diseño la elaboración de los diagramas de clases de diseño juega un papel fundamental, estos muestran las clases finales para la realización de los casos de uso modelados con anterioridad.

Los diagramas de clases de diseño muestran las clases con sus atributos y métodos y la forma en que se relacionan entre sí. En este flujo también se realizan los diagramas de interacción que muestran la comunicación y las relaciones entre los objetos, con el objetivo de dar cumplimiento a los requerimientos.

Además, el diagrama de clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Permite visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de convencimiento (22). En la Figura 4 se muestra el diagrama de clases al sistema desarrollado agrupado en paquetes de acuerdo a la arquitectura y en la Tabla 12 se describen las clases.

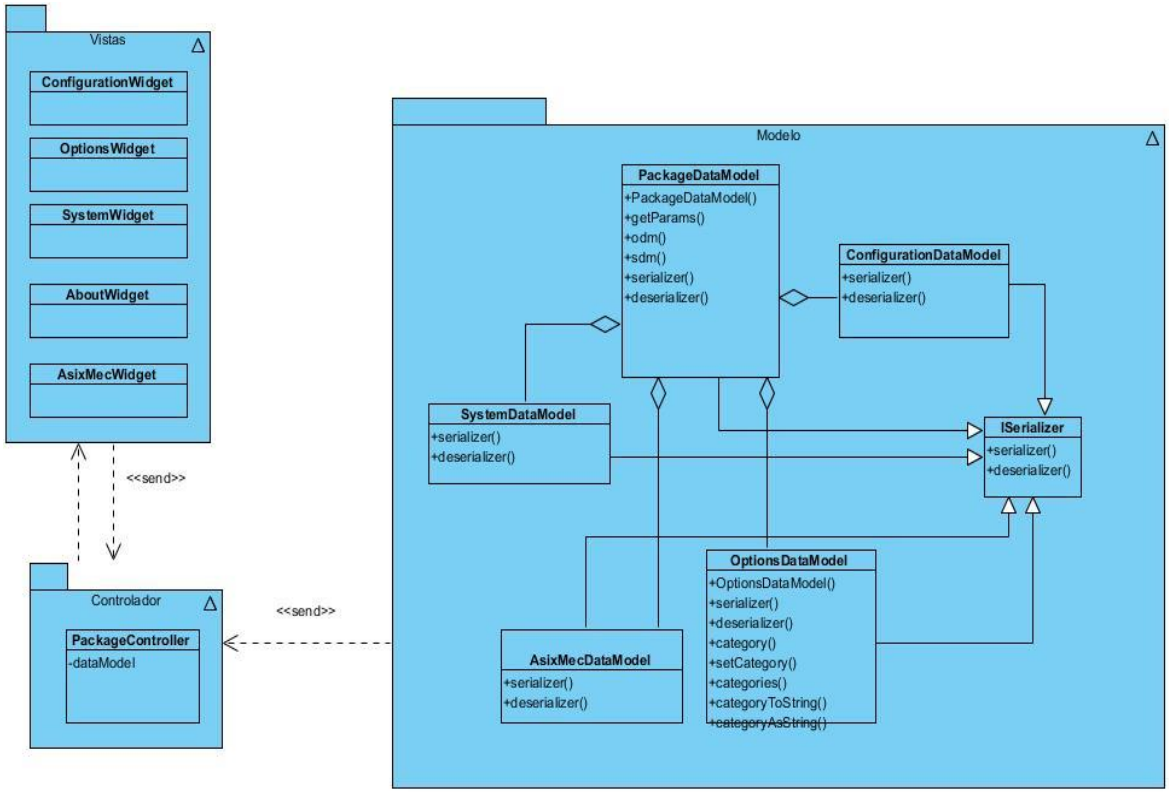

Figura 4. Diagrama de clases.

Tabla 12. Descripción de las clases.

Clases	Descripción	Paquete
ConfigurationWidget	Esta clase contiene las configuraciones requeridas para la creación del paquete.	Vistas
OptionsWidget	Esta clase contiene las configuraciones opcionales para la creación del paquete.	Vistas
SystemWidget	Esta clase contiene las configuraciones del sistema para la creación del paquete.	Vistas
AsiXMecWidget	Esta clase contiene las configuraciones específicas del proyecto AsiXMec para la creación del paquete.	Vistas
PackegeController	Esta es clase encargada de controlar la creación del paquete.	Controlador
ConfigurationDataModel	Esta clase es la abstracción que mapea los datos de las	Modelo

	configuraciones requeridas para la creación del paquete.	
OptionsDataModel	Esta clase es la abstracción que mapea los datos de las configuraciones opcionales para la creación del paquete.	Modelo
SystemDataModel	Esta clase es la abstracción que mapea los datos de las configuraciones del sistema para la creación del paquete	Modelo
AsiXMecDataModel	Esta clase es la abstracción que mapea los datos de las configuraciones específicas del proyecto AsiXMec para la creación del paquete.	Modelo
PackegeDataModel	Esta clase es la abstracción que mapea los datos para la creación del paquete.	Modelo

Para lograr una asignación correcta de las responsabilidades en las clases representadas en el diagrama de paquetes y un diseño adecuado de las relaciones entre sus objetos, es necesario tener en cuenta el uso de los patrones de diseño, garantizando organización y uniformidad en la implementación del sistema.
Visitor: es un patrón de comportamiento que permite definir una operación sobre objetos de una jerarquía de clases sin modificar las clases sobre las que opera. Representa una operación que se realiza sobre los elementos que conforman la estructura de un objeto. A continuación, como se puede apreciar en la Figura 5 se muestra el patrón visitor.

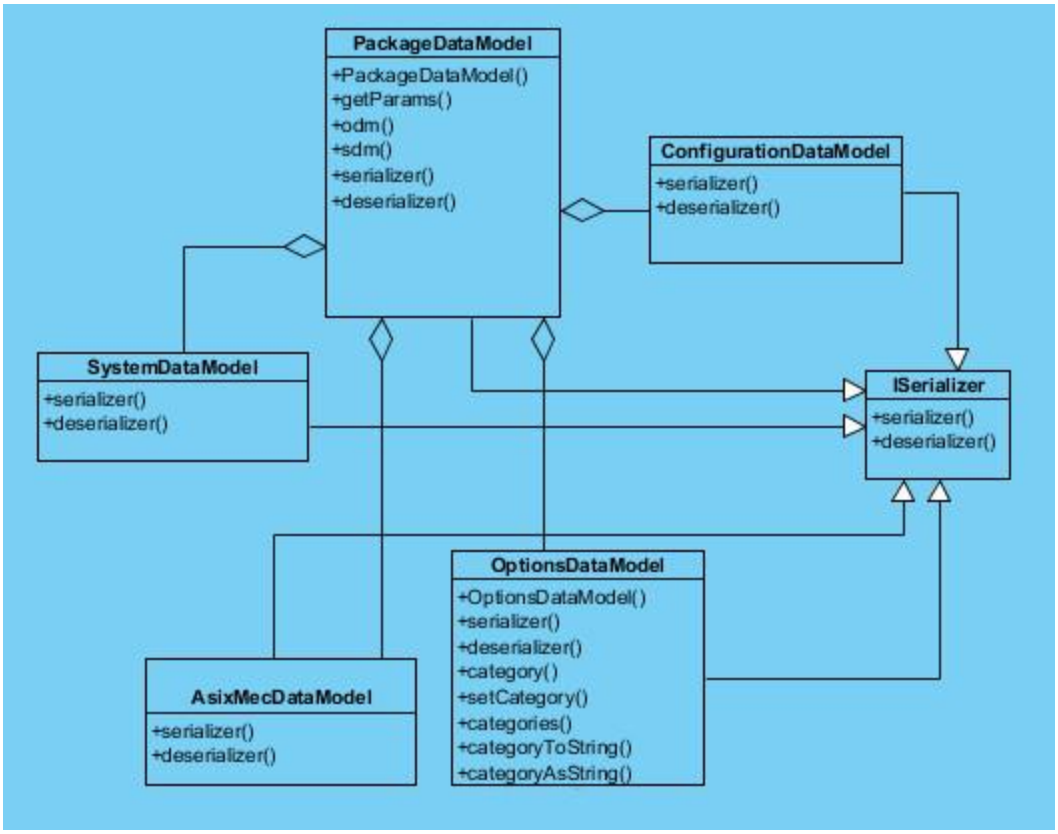

Figura 5. Patrón visitor en el código.

2.6. Consideraciones parciales

Luego de analizar y diseñar el sistema, teniendo en cuenta las características y cualidades que debe cumplir el mismo según las necesidades del cliente, se cuenta con una arquitectura en capas en la que se organiza el código de la aplicación. La aplicación de técnicas para la captura de requisitos posibilitó identificar las funcionalidades del instalador a partir de las necesidades del cliente. La generación de los diagramas correspondientes al diseño de la aplicación facilitó la comprensión de las relaciones entre las clases presentes en el modelo.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS

Partiendo del resultado del análisis y diseño, en el presente capítulo se generan los productos de trabajo referentes a la fase de implementación y prueba del software. Como principales elementos se definen los diagramas de componentes y de despliegue. Se especifican los casos de pruebas aplicados a la solución desarrollada para validar su correcto funcionamiento.

3.1. Implementación

El resultado principal de la implementación, es la obtención de componentes, dentro de los que se incluyen ficheros, ejecutables y las dependencias existentes entre estos. Además, este flujo especifica cómo van a estar ubicadas físicamente las distintas partes del sistema. En los casos necesarios se incluye también los protocolos que se emplearán para la comunicación entre los componentes.

3.1.1. Diagrama de componentes

Un componente representa una parte física del sistema, por ejemplo, una biblioteca de clases, un ejecutable, una tabla, que engloba la implementación de un grupo de clases del diseño. Cada componente define una interfaz que describe su funcionalidad y forma de empleo. El diagrama de componentes, permite conocer a los desarrolladores y clientes la estructura física que tiene el sistema y cómo se relacionan sus partes. A continuación, se presenta el diagrama de componentes (ver Figura 6) del sistema propuesto. En el diagrama se incluye un paquete con los componentes ejecutables que son necesarios para la aplicación. Luego como se puede apreciar en Tabla 13 se describen ocho componentes que son necesarios para la ejecución de la aplicación.

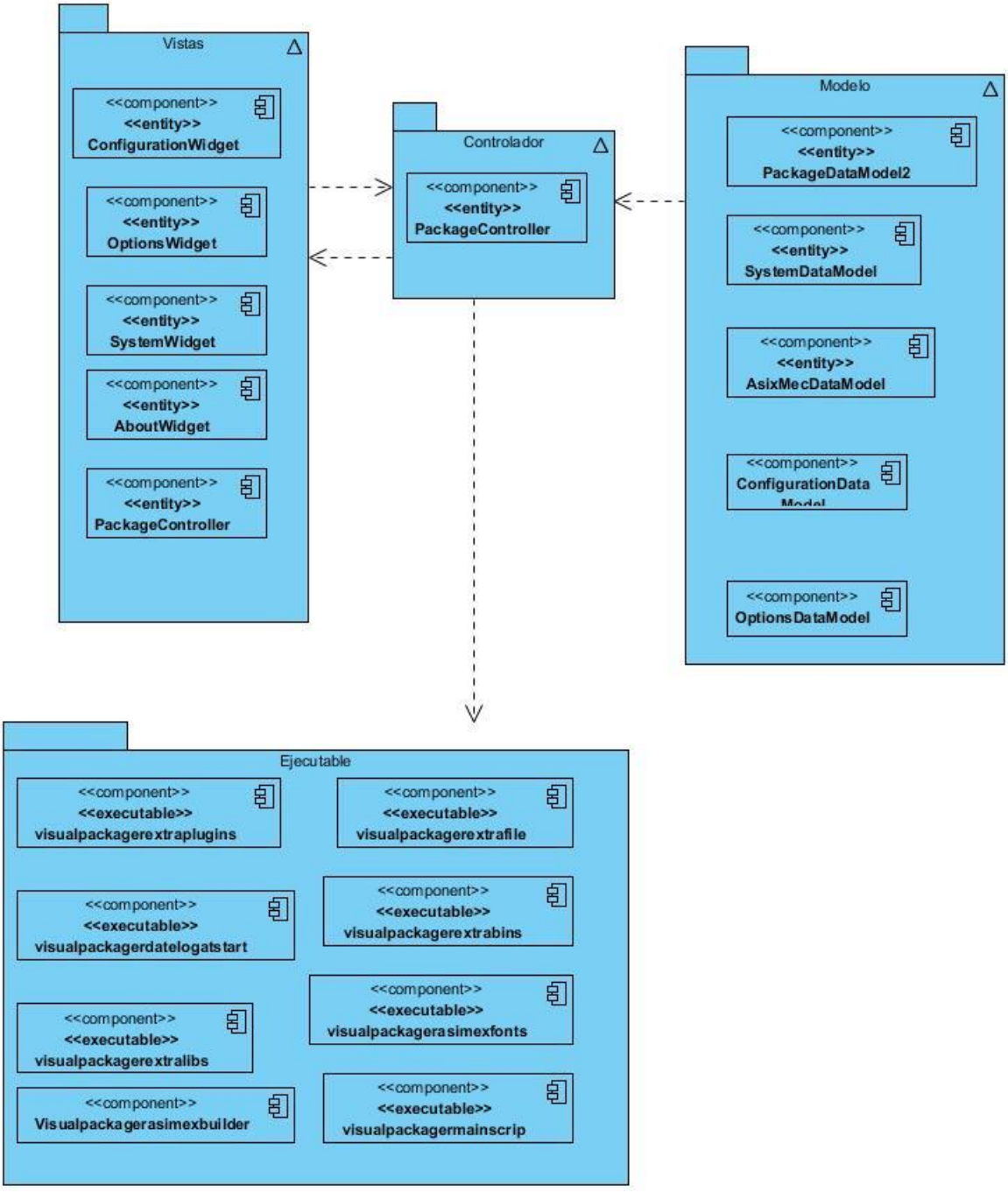

Figura 6. Diagrama de componentes.

Tabla 13. Descripción de los componentes.

Componentes	Descripción
Visualpackagerextraplugins	Ejecutable en bash para copiar los complementos auxiliares.
Visualpackagerextralibs	Ejecutable en bash para copiar las bibliotecas auxiliares.
Visualpackagerextrafile	Ejecutable en bash para copiar los archivos.
Visualpackagerextrabins	Ejecutable en bash para copiar los binarios auxiliares.
Visualpackagerdatelogatstart	Ejecutable en bash para marcar el inicio de los registros.
Visualpackagerasimexfonts	Ejecutable en bash para copiar las fuentes del proyecto específico AsiXMec.
Visualpackagermainscrip	Ejecutable principal en bash que ordena la ejecución de los demás ejecutables y construye el paquete utilizando la herramienta dpkg.
Visualpackagerasimexbuilder	Ejecutable en bash que ordena la compilación del código fuente del proyecto específico AsiXMec.

3.2. Pruebas

Una vez implementado los casos de uso definidos para el sistema, se transita hacia la fase de pruebas con el propósito de obtener una solución libre de errores. Durante su desarrollo, la aplicación fue sometida a un conjunto de casos de pruebas, a partir de las iteraciones definidas, con el objetivo de garantizar su buen funcionamiento.

Las pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada. Son un conjunto de situaciones o condiciones ante las cuales un programa debe responder satisfactoriamente para que pueda afirmarse que cumple con sus especificaciones u objetivos. Es una actividad más en el proceso de control de calidad. Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo. Existen distintos modelos de desarrollo de software, así como modelos de pruebas (23).

Según la metodología variante AUP-UCI en la disciplina de pruebas internas se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias. Se realizaron pruebas funcionales con la técnica de caja negra.

3.2.1. Diseño de Casos de Prueba

Los casos de prueba son la forma de verificar las diversas funcionalidades existentes en un producto de software descritas en el formato de los Casos de Uso. Estos se realizan con el objetivo de conseguir un margen de confianza aceptable de que serán encontrados todos los defectos existentes sin consumir una cantidad excesiva de recursos. A continuación, se ilustra el diseño de caso de prueba del caso de uso Administrar configuración del paquete de instalación, se puede apreciar las descripción de las variables en Tabla 14 y los escenarios de prueba en la

Tabla 15. Durante la realización de pruebas se utilizó la técnica de diseño de caja negra con el método de prueba partición de clases de equivalencia. El resto de los casos de prueba se pueden encontrar en el ANEXO 2: Casos de prueba

Tabla 14. Descripción de variables del DCP- Administrar configuración del paquete de instalación.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
V1	Nombre del paquete.	Campo de texto	No	Permite todo tipo de carácter excepto el espacio.
V2	Versión del software.	Campo de texto	No	Permite todo tipo de carácter.
V3	Descripción del software.	Campo de texto	No	Permite todo tipo de carácter.
V4	Ejecutable principal del paquete.	Cargar fichero	No	Permite cargar una ruta válida de un fichero y debe existir en la computadora.
V5	Ruta de generado	Cargar directorio	No	Permite cargar una ruta válida de un directorio y debe existir en la computadora.

Tabla 15. DCP- Administrar configuración del paquete de instalación. [4]

Escenario	Descripción	V1	V2	V3	V4	V5	Respuesta del sistema	Flujo central
EC 1.1 Introducir datos correctos en los campos.	La aplicación debe permitir la creación del paquete.	V	V	V	V	V	Se crea el instalador de paquetes.	- Interfaz principal de la aplicación en *Configuración*. - Introducir los datos. - Seleccionar el botón *generar*.
		libreoffice-base	1.5	Este paquete tiene la base de datos.	/etc/ libreoffice/ soffice.sh	/home/ trabajo		
		V	V	V	V	V		
		vlc	2.1.4	Este paquete es para reproducir videos.	/usr/lib/ vlc/vlc-cache-gen	/home/ trabajo		
EC 1.2 Introducir datos incorrectos en los campos.	La aplicación no debe permitir la creación del paquete.	I	V	V	V	V	La aplicación muestra un mensaje de error y no se crea el instalador de paquetes.	- Interfaz principal de la aplicación en *Configuración*. - Introducir los datos.
		pid gin	2.0	Este paquete es un cliente de mensajería	/etc/pidgin/	/home/ trabajo		

[4] **Nota:** V es válido, I es inválido.

3.2.2. Validación de la propuesta

Con el objetivo de validar la propuesta de solución, al mismo se le realizaron diversas pruebas una vez ya concluida la etapa de desarrollo con el fin de justificar su valía. Se realizaron pruebas funcionales y de aceptación con la técnica de diseño de prueba caja negra a nivel de interfaz. Además, se utilizó el método de partición de equivalencia para examinar los valores válidos e inválidos de las entradas existentes en la aplicación.

Se diseñaron 11 casos de pruebas, divididos en 4 iteraciones. En la primera iteración se encontraron 9 no conformidades, de estas: 5 de validación, 2 de funcionalidad y 2 de correspondencia con el DCP. En la segunda iteración se detectaron 6 no conformidades, de estas: 3 de validación, 1 de excepción y 2 de ortografía. En la tercera iteración se encontraron 2 no conformidades, estas: 1 de validación y 1 de funcionalidad. En la última iteración no se detectaron fallos en la aplicación. Los resultados de las pruebas se presentan en la Tabla 16 y en el Gráfico 1 y Gráfico 2.

Tabla 16. No conformidades detectadas en cada iteración.

Primera Iteración	Segunda Iteración	Tercera Iteración
Permite crear un paquete con una ruta incorrecta. (validación)	No se muestra el mensaje de error dejando el campo (*Ejecutable principal del paquete*) vacío. (validación) (no conformidad pendiente)	Permite crear un paquete con una ruta incorrecta. (validación)
No se muestra el mensaje de error dejando el campo (*Descripción del software*) vacío. (validación)	Permite introducir una dirección inválida en el campo *Página Oficial* de la sesión *Opciones*. (validación)	No se crea el paquete con arquitectura de 64bit cuando es especificado previamente. (funcionalidad)
No se muestra el mensaje de error dejando el campo (*Ejecutable principal del paquete*) vacío. (validación)	En el campo *Pro file* se permite extensiones de archivos que no se corresponde. (validación)	
No se muestra el mensaje de error cuando	Se muestra un mensaje de	

se introduce *espacio* en el campo *Nombre del paquete* (validación)	excepción cuando se selecciona el botón *generar* (excepción**)**	
No se muestra el mensaje de error dejando el campo (*Ruta de generado*) vacío. (validación)	Le falta la tilde a la palabra versión. (ortografía)	
No se puede añadir una dependencia. (funcionalidad)	Está mal escrita la palabra configuración. (ortografía**)**	
No se puede seleccionar el checkbox del campo *Dependencia*. (funcionalidad)		
En la aplicación dice versión del paquete y en el DCP versión del software. (correspondencia)		
En el flujo del central del DCP dice Configuración y en la aplicación dice Configuración del sistema. (correspondencia)		

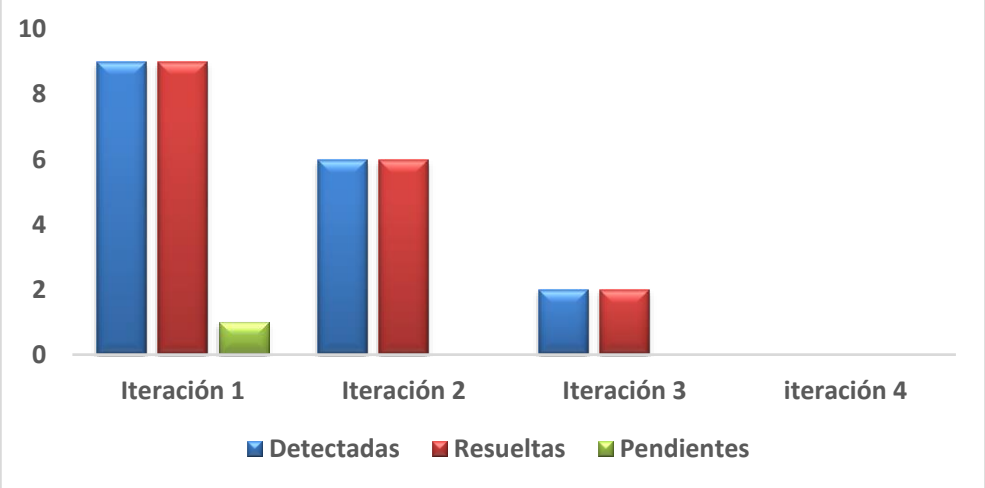

Gráfico 1. No conformidades por iteración.

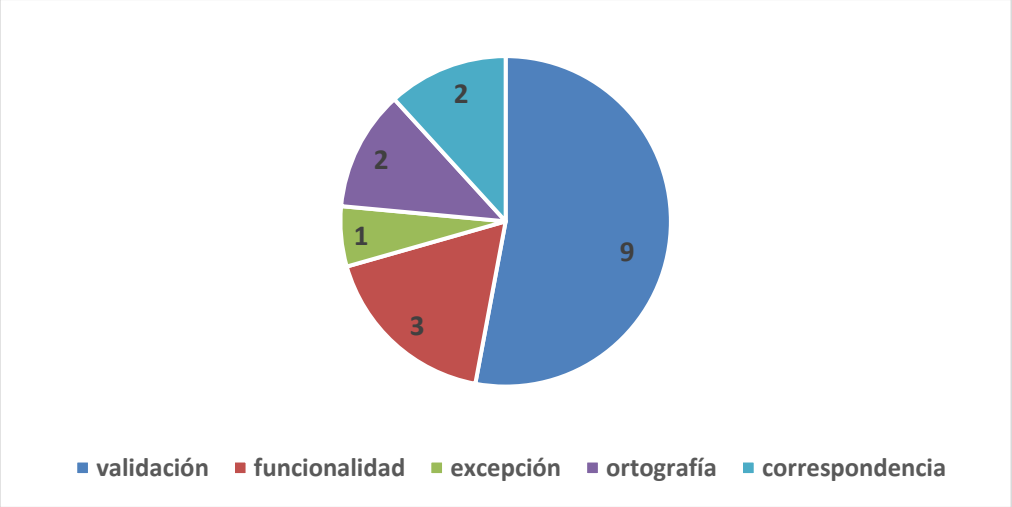

Gráfico 2. Tipos de no conformidades detectadas.

3.3. Creación de paquetes de instalación para AsiXMec con el Empaquetador Visual

De forma general se puede crear un paquete de instalación desde la aplicación Empaquetador Visual (*visualpackager)* se deben seguir los siguientes pasos:

1. Ejecutar la aplicación *visualpackager.*
2. Introducir los datos en la opción *Configuración*.
3. En caso que se desee añadir otros parámetros se puede acceder a las demás opciones.
4. Seleccionar la opción *Generar*.

La creación de paquetes de instalación para AsiXMec se puede realizar de formas:

1. Compilar el código fuente del producto, luego llenar los campos requeridos en las pestañas de *Configuración, Opciones* y las restantes configuraciones de AsiXMec.
2. Tener previamente el producto compilado e introducir los datos en los campos en *Configuración, Opciones* y las restantes configuraciones de AsiXMec.

Para crear un paquete de instalación es necesario seleccionar la opción *Configuración* e introducir el nombre del paquete, la versión del software, la descripción del software, el ejecutable del software y la

ruta de generado. Un vez que los datos sean introducidos se debe seleccionar la opción *Generar* (ver Figura 7).

Antes de generar la creación del paquete de instalación, si el usuario desea especificar otras configuraciones debe seleccionar *Opciones* (ver Figura 8). Para el caso de un producto AsiXMec se deben especificar los siguientes elementos:

Dependencias: *qt5-default, qttools5-dev, qttools5-dev-tools, libqt5svg5-dev, ftgl-dev, libxmu-dev, tcl-dev, freeglut3-dev, liblog4cxx10-dev, libeigen3-dev, libqt5sql5-psql, libboost-graph-dev, qttranslations5-l10n, libqt5webkit5-dev, libqt5webkit5, libqt5network5.*

Complementos adicionales: *base-plugin, break-plugin, cota-plugin, graph-based-face-identification-plugin, notes-plugin, smart-sketch-plugin, solver-2d-plugin, std-data-exchange-plugin.*

Fuente: *arial.ttf*.

En la interfaz AsiXMec (ver Figura 9), si el producto no ha sido compilado, se deben introducir en los campos la dirección del *.pro* y el directorio de compilación, luego seleccionar la opción *Compilar*. Una vez que el proyecto es compilado se completan los restantes campos.

La herramienta cuenta con otras opciones, para ello se debe seleccionar *Sistema* (ver Figura 10) donde:

1. si se selecciona la opción *Aplicación de consola,* la aplicación no se muestra en el menú del sistema y será vía comando.
2. si se selecciona la opción *Ejecutar como administrador*, la aplicación requiere de privilegios especiales y de contraseña *root*.

Figura 7. Interfaz Configuración

Figura 8. Interfaz Opciones

Figura 9. Interfaz AsiXMec

Figura 10. Interfaz Sistema

3.4. Análisis de los resultados

Para que la aplicación funcione correctamente es necesario que el sistema operativo tenga la biblioteca libqtwidget (a partir de su versión 5.0) y el paquete dpkg debe estar instalado.

Se realizaron además pruebas de sistema y de aceptación para evaluar el funcionamiento de la aplicación como un todo y observar el grado de aceptación entre los usuarios finales. El tiempo de respuesta de las transacciones fue de 10 000 milisegundos. Se probó el software en sistemas operativos de las distribuciones de Ubuntu 14.04 o superior; Debian 8 o Nova 5.0 y se concluye que la propuesta es aceptable, además el uso de la misma debe de ser en computadoras con 2.0 GB Mínimo de RAM y un procesador: Intel Core i3 a 3.3 GHz o superior.

Para la instalación de la aplicación Empaquetador Visual (visualpackager) se deben seguir los siguientes pasos:

Vía 1:

1. Seleccionar la aplicación *visualpackager* y dar doble clic en esta.
2. Seleccionar la opción *instalar paquete*.

Vía 2:

1. Abrir una terminal.
2. Tener permiso de administración.
3. Introducir la línea de comando *dpkg –i /dirección de la aplicación* y dar *enter*.

3.5. Consideraciones parciales

Tras definir las características de implementación y pruebas de la propuesta de solución, se arrojan las siguientes conclusiones:

- ✓ La modelación del sistema permitió obtener la vista estática del mismo a través de los diagramas de componentes y el diagrama de despliegue que indica la situación física de los componentes lógicos desarrollados.
- ✓ Con el objetivo de comprobar el correcto comportamiento de los requerimientos del sistema se realizaron las pruebas para detectar en tiempo defectos que pudiera presentar el sistema; y se validó la solución propuesta a través de las pruebas de sistema y de aceptación las que permitieron evaluar el funcionamiento del sistema.

CONCLUSIONES

Con la realización de este trabajo, se desarrolló una aplicación de escritorio que facilita el proceso de creación de instaladores del proyecto DISEM en GNU/Linux. De esta forma se da cumplimiento al objetivo propuesto al inicio de la investigación, además:

- Las funcionalidades de guardar y cargar la configuración permiten volver a generar los instaladores en un tiempo de configuración no mayor a un minuto.
- Los instaladores generados gestionan las dependencias de las bibliotecas externas a partir de la configuración en la aplicación desarrollada, lo que facilita el despliegue de los productos que utilicen la herramienta propuesta.
- Se disminuye la curva de aprendizaje para la generación de instaladores por parte de los nuevos desarrolladores, al interactuar con una interfaz visual y no directamente con la línea de comandos.
- Como resultado adicional, la aplicación desarrollada es funcional para generar también instaladores no relacionados con el proyecto DISEM.

RECOMENDACIONES

A partir del estudio realizado en la presente investigación se hacen las siguientes recomendaciones:

- Ejecutar el proceso de generación del paquete en otro hilo que no sea el principal de la aplicación.
- Adicionar soporte para el sistema operativo Windows.

BIBLIOGRAFÍA

1. Lasso Mendez, Iván. Proyecto Autodidacta. [En línea] [Citado el: 03 de noviembre de 2015.] http://www.proyectoautodidacta.com/2008/05/09/%C2%BFque-es-un-instalador/.

2. Real Academia Española. 23. edición del Diccionario de la Lengua Española. [En línea] [Citado el: 03 de noviembre de 2015.] http://dle.rae.es/?id=LmaRZha.

3. Hohmann, Luke. Beyond software architecture: creating and sustaining winning solutions. s.l. : Addison-Wesley Longman Publishing Co., Inc., 2003.

4. Microsoft. [En línea] [Citado el: 05 de noviembre de 2015.] https://www.microsoft.com/es-mx/howtotell/empaquetadosotfware/..

5. GNU Operating System. Bash - GNU Project. [En línea] [Citado el: 6 de noviembre de 2015.] https://www.gnu.org/software/bash/.

6. Tanenbaum, Andrew. Sistemas Operativos Modernos. segunda. México : Pearson Educación, 2003.

7. OpenTheFile. DEB extensión de archivo. [En línea] [Citado el: 8 de noviembre de 2015.] http://www.openthefile.net/es/extension/deb.

8. Bravo, Thomas. InstallShield. [En línea] [Citado el: 12 de noviembre de 2015.] http://consumer.installshield.com/.

9. Softonic. [En línea] [Citado el: 24 de noviembre de 2015.] http://setup-factory.softonic.com/..

10. InstallAnywhere: Multiplatform installs from a single project. [En línea] [Citado el: 24 de noviembre de 2015.] http://www.flexerasoftware.com/producer/products/software-installation/installanywhere/.

11. Hess, Antiun. manpages.spotlynx. [En línea] [Citado el: 15 de noviembre de 2015.] http://www.manpages.spotlynx.com/gnu_linux/man/debhelper.7.

12. El libro del administrador de Debian. [En línea] https://debian-handbook.info/browse/es-ES/stable/sect.manipulating-packages-with-dpkg.html.

13. Ubuntu. [En línea] http://manpages.ubuntu.com/manpages/saucy/es/man1/fakeroot-tcp.1.html.

14. qt.io. [En línea] https://www.qt.io/ide/.

15. Carrillo Pérez, Saías. METODOLOGIA DE DESARROLLO DEL SOFTWARE. 2008.

16. Krutchen, Philippe. The Rational Unified Process: an introduction. Third Edition. 3rd. s.l. : Addison-Wesley Professional, 2004.

17. Guillermo Lafuente, Javier . UML Unified Modeling Languaje. [En línea] [Citado el: 15 de enero de 2013.] http://gidis.ing.unlpam.edu.ar/personas/glafuente/uml/uml.html....

18. El lenguaje C++. [En línea] http://www.zator.com/Cpp/E1_2.htm..

19. Olivera, Angel. Requerimientos funcionales y no funcionales. [En línea] 2010. http://www.slideshare.net/lilyPacheco7/arquitectura-de-software-13925226.

20. SoftQaNetwork. [En línea] [Citado el: 26 de 03 de 2014.] http://www.softqanetwork.com/requisitosno-funcionales-nfr..

21. Jacobson, Ivar , Booch, Grady y Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software.* [ed.] Andrés Otero. [trad.] Salvador Sánchez, y otros. s.l. : Pearson Educacion S.A., 2000. ISBN: 84-7829-036-2.

22. [En línea] gdamar877.blogspot.com/2009/05/expocicion.html.

23. prezi. [En línea] [Citado el: 3 de abril de 2014.] http://prezi.com/fdkxfkwcrb5o/copy-of-mapaconceptual-de-tecnicas-de-pruebas-y-mantenimiento-de-software/..

ANEXOS

ANEXO 1: Descripción de casos de uso

Tabla 17: Configurar sistema.

Caso de Uso	Administrar opciones del sistema.	
Objetivo	Administrar la configuración del sistema.	
Actores	Usuario	
Resumen	El caso de uso se inicia cuando el usuario accede a la opción" sistema", este puede seleccionar la configuración. El caso de uso finaliza.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	-	
Postcondiciones	Se realiza la configuración del sistema.	
Flujo de eventos		
Flujo básico Configurar sistema		
	Actor	**Sistema**
1.	Selecciona la opción "Sistema" .	
2.		Muestra los campos correspondientes: ✓ Aplicación de consola. ✓ Ejecutar como administrador.
3.	Selecciona la opción deseada.	
4.		Finaliza el caso de uso.
Relaciones	**CU incluidos**	No aplica
	CU extendidos	No aplica
Requisitos no funcionales	No aplica	
Asuntos pendientes	No aplica	

Tabla 18: Administrar categoría especifica del proyecto.

Caso de Uso	Administrar categoría especifica del proyecto.
Objetivo	Administrar la categoría especifica del proyecto AsiXMec.
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario accede a la opción AsiXMec, el sistema permite introducir los datos necesarios para la configuración de AsiXMec, el usuario del sistema introduce los datos necesarios, el caso de uso finaliza.
Complejidad	Alta
Prioridad	Alta
Precondiciones	-
Postcondiciones	Se realiza la administración de la categoría especifica del proyecto AsiXMec
Flujo de eventos	

Flujo básico Configurar AsiXMec		
	Actor	**Sistema**
1.	Selecciona la opción “AsiXMec”.	
2.		Muestra los datos a ser llenados: ✓ Librería OCE. ✓ Fuentes (ver CU Gestionar Fuentes). ✓ Main folder. ✓ Output ✓ Source ✓ Build form source ✓ Install depends automatically
3.	Inserta los datos pertinentes a dichos campos.	
4.		Finaliza el caso de uso.
Relaciones	**CU incluidos**	**No aplica**
	CU extendidos	**-CU Administrar Fuentes.**
Requisitos no funcionales	**No aplica**	
Asuntos pendientes	**No aplica**	

Tabla 19: Gestionar dependencias.

Caso de Uso	Gestionar dependencias	
Objetivo	Gestionar las dependencias con otras aplicaciones	
Actores	Usuario	
Resumen	El caso de uso se inicia cuando el usuario accede a la opción “Dependencias”, el sistema permite administrar las dependencias del paquete de instalación, el caso de uso finaliza.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	-	
Postcondiciones	Se realiza la gestión de las dependencias.	
Flujo de eventos		
Flujo básico Gestionar dependencias		
	Actor	**Sistema**
1.	Selecciona la opción “Opciones”.	
2.		Muestra los datos a ser llenados: ✓ Dependencia. ✓ Categoría. ✓ Arquitectura. ✓ Autor. ✓ Página oficial. ✓ Nombre del programa. ✓ Imágenes del programa. ✓ Binarios Auxiliares. (ver CU Administrar binarios auxiliares) ✓ Iconos del programa.

		✓ Bibliotecas adicionales. (ver CU Administrar bibliotecas adicionales) ✓ Complementos adicionales (ver CU Administrar Complementos adicionales). ✓ Traducciones. ✓ Post install script.
3.	Selecciona el campo “Dependencia”.	
4.		Muestra las opciones que puede realizar: ✓ Agregar dependencia (ver Sección 1). ✓ Modificar dependencia (ver Sección 2). ✓ Eliminar dependencia (ver Sección 3).
5.		Finaliza el caso de uso.
Sección 1: “Agregar dependencia”		
Flujo básico		
	Actor	**Sistema**
1.	Escribe el nombre de la dependencia y da clic en el botón “Agregar”.	
2.		Muestra el nombre de la dependencia recientemente agregada.
Sección 2: “Modificar dependencia”		
Flujo básico		
	Actor	**Sistema**
1.	Selecciona la dependencia a modificar y al mismo tiempo modifica el nombre de la dependencia seleccionada.	
2.		Muestra la dependencia ya modificada con las restantes dependencias en caso de existir otras.
Sección 3: “Eliminar dependencia”		
Flujo básico		
1.	Selecciona la dependencia a eliminar y presiona el botón “Quitar”.	
2.		Muestra las dependencias restantes en caso de existir.
Relaciones	**CU incluidos**	No aplica
	CU extendidos	No aplica
Requisitos no funcionales	No aplica	
Asuntos pendientes	No aplica	

Tabla 20: Administrar ejecutables auxiliares.

Caso de Uso	Gestionar ejecutables auxiliares.
Objetivo	Gestionar los ejecutables auxiliares necesarios para el creador de paquetes.
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario del sistema accede a la opción “binarios

	auxiliares", el sistema permite administrar los binarios auxiliares del paquete de instalación, el caso de uso finaliza.
Complejidad	Alta
Prioridad	Alta
Precondiciones	
Postcondiciones	

Flujo de eventos		
Flujo básico Gestionar dependencias		
	Actor	**Sistema**
1.	Selecciona la opción "Opciones".	
2.		Muestra los datos a ser llenados: ✓ Dependencia. ✓ Categoría. ✓ Arquitectura. ✓ Autor. ✓ Página oficial. ✓ Nombre del programa. ✓ Imágenes del programa. ✓ Binarios Auxiliares. ✓ Iconos del programa. ✓ Bibliotecas adicionales. (ver CU Administrar Bibliotecas adicionales) ✓ Complementos adicionales (ver CU Administrar complementos adicionales). ✓ Traducciones. ✓ Post install script.
3.	Selecciona el campo "Binarios auxiliares".	
4.		Muestra una ventana emergente con las opciones que puede realizar: ✓ Agregar binarios auxiliares (ver Sección 1). ✓ Modificar binarios auxiliares (ver Sección 2). ✓ Eliminar binarios auxiliares (ver Sección 3).
5.		Finaliza el caso de uso.
Sección 1: "Agregar binarios auxiliares"		
Flujo básico		
	Actor	**Sistema**
1.	Selecciona el botón "Adicionar".	
2.		Muestra una ventana emergente para poder seleccionar el binario auxiliar.
Flujos alternos		
2a. Cancelar la acción de agregar binarios auxiliares.		
	Actor	**Sistema**
1.	Selecciona el botón "Cancelar".	
2.		Vuelve a la pantalla de adicionar binarios auxiliares.
Sección 2: "Modificar binarios auxiliares"		

Flujo básico		
	Actor	**Sistema**
1.	Selecciona el binario auxiliar a modificar y presiona el botón “Cambiar”.	
2.		Muestra una ventana emergente para poder seleccionar el binario auxiliar que va a sustituir al anterior.
Flujos alternos		
2a. Cancelar la acción de modificar binarios auxiliares.		
	Actor	**Sistema**
1.	Presiona el botón “Cancelar”.	
2.		Vuelve a la pantalla de adicionar binarios auxiliares, con los binarios auxiliares ya existentes.
Sección 3: “Eliminar binarios auxiliares”		
Flujo básico		
1.	Selecciona el binario auxiliar a eliminar y presiona el botón “Eliminar”.	
2.		Muestra los binarios auxiliares restantes en caso de existir.

Relaciones	**CU incluidos**	No aplica
	CU extendidos	No aplica
Requisitos no funcionales	No aplica	
Asuntos pendientes	No aplica	

Tabla 21: Gestionar bibliotecas adicionales.

Caso de Uso	Gestionar bibliotecas adicionales
Objetivo	Gestionar las bibliotecas adicionales necesarias para el creador de paquetes.
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario del sistema accede a la opción “bibliotecas adicionales”, el sistema permite administrar las bibliotecas adicionales del paquete de instalación, el caso de uso termina.
Complejidad	Alta
Prioridad	Alta
Precondiciones	-
Postcondiciones	Se realiza la gestión de bibliotecas adicionales.
Flujo de eventos	
Flujo básico Administrar ruta de librería	

	Actor	**Sistema**
1.	Selecciona la opción “Opciones”.	
2.		Muestra los datos a ser llenados: ✓ Dependencia. ✓ Categoría. ✓ Arquitectura. ✓ Autor. ✓ Página oficial. ✓ Nombre del programa. ✓ Imágenes del programa.

		✓ Binarios Auxiliares. (ver CU Administrar binarios auxiliares) ✓ Iconos del programa. ✓ Bibliotecas adicionales. ✓ Complementos adicionales (ver CU Administrar complementos adicionales). ✓ Traducciones. ✓ Post install script.
3.	Selecciona el campo “Bibliotecas auxiliares”.	
4.		Muestra una ventana emergente con las opciones que puede realizar: ✓ Agregar bibliotecas auxiliares (ver Sección 1). ✓ Modificar bibliotecas adicionales (ver Sección 2). ✓ Eliminar bibliotecas adicionales (ver Sección 3).
5.		Finaliza el caso de uso.
Sección 1: “Agregar bibliotecas auxiliares”		
Flujo básico		
	Actor	**Sistema**
1.	Selecciona el botón “Adicionar”.	
2.		Muestra una ventana emergente para poder seleccionar una biblioteca adicional.
Flujos alternos		
2a. Cancelar la acción de agregar bibliotecas auxiliares.		
	Actor	**Sistema**
1.	Selecciona el botón “Cancelar”.	
2.		Vuelve a la pantalla de adicionar bibliotecas auxiliares.
Sección 2: “Modificar bibliotecas auxiliares”		
Flujo básico		
	Actor	**Sistema**
1.	Selecciona la biblioteca adicional a modificar y presiona el botón “Cambiar”.	
2.		Muestra una ventana emergente para poder seleccionar la biblioteca adicional que va a sustituir a la anterior.
Flujos alternos		
2a. Cancelar la acción de modificar bibliotecas auxiliares.		
	Actor	**Sistema**
1.	Presiona el botón “Cancelar”.	
2.		Vuelve a la pantalla de adicionar bibliotecas adicionales, con las bibliotecas adicionales ya existentes.
Sección 3: “Eliminar bibliotecas auxiliares”		
Flujo básico		
1.	Selecciona la biblioteca adicional a eliminar y presiona el botón “Eliminar”.	
2.		Muestra las bibliotecas adicionales restantes en caso de existir.
Relaciones	**CU incluidos**	No aplica

	CU extendidos	No aplica
Requisitos no funcionales	No aplica	
Asuntos pendientes	No aplica	

Tabla 22: Gestionar complementos adicionales.

Caso de Uso	Gestionar complementos adicionales.	
Objetivo	Gestionar los complementos adicionales necesarios para el creador de paquetes.	
Actores	Usuario	
Resumen	El caso de uso se inicia cuando el usuario del sistema accede a la opción “complementos adicionales”, el sistema permite administrar los complementos adicionales del paquete de instalación, el caso de uso termina.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	-	
Postcondiciones	Se realiza la gestión de complementos adicionales.	
Flujo de eventos		
Flujo básico Administrar ruta de librería		
	Actor	**Sistema**
1.	Selecciona la opción “Opciones”.	
2.		Muestra los datos a ser llenados: ✓ Dependencia. ✓ Categoría. ✓ Arquitectura. ✓ Autor. ✓ Página oficial. ✓ Nombre del programa. ✓ Imágenes del programa. ✓ Binarios Auxiliares. (ver CU Administrar binarios auxiliares) ✓ Iconos del programa. ✓ Bibliotecas adicionales (ver CU Administrar bibliotecas adicionales) ✓ Complementos adicionales. ✓ Traducciones. ✓ Post install script.
3.	Selecciona el campo “Complementos adicionales”.	
4.		Muestra una ventana emergente con las opciones que puede realizar: ✓ Agregar complementos adicionales (ver Sección 1). ✓ Modificar complementos adicionales (ver Sección 2). ✓ Eliminar complementos adicionales (ver Sección 3).
5.		Finaliza el caso de uso.

Sección 1: “Agregar complemento adicionales”		
Flujo básico		
	Actor	**Sistema**
1.	Selecciona el botón “Adicionar”.	
2.		Muestra una ventana emergente para poder seleccionar los complementos adicionales.
Flujos alternos		
2a. Cancelar la acción de agregar complementos adicionales.		
	Actor	**Sistema**
1.	Selecciona el botón “Cancelar”.	
2.		Vuelve a la pantalla de adicionar complementos adicionales.
Sección 2: “Modificar complementos adicionales”		
Flujo básico		
	Actor	**Sistema**
1.	Selecciona el complemento adicional a modificar y presiona el botón “Cambiar”.	
2.		Muestra una ventana emergente para poder seleccionar el complemento adicional que va a sustituir al anterior.
Flujos alternos		
2a. Cancelar la acción de modificar complementos adicionales.		
	Actor	**Sistema**
1.	Presiona el botón “Cancelar”.	
2.		Vuelve a la pantalla de adicionar complementos adicionales, con los complementos adicionales ya existentes.
Sección 3: “Eliminar complementos adicionales”		
Flujo básico		
1.	Selecciona el complemento adicional a eliminar y presiona el botón “Eliminar”.	
2.		Muestra los complementos adicionales restantes en caso de existir.
Relaciones	**CU incluidos**	No aplica
	CU extendidos	No aplica
Requisitos no funcionales	No aplica	
Asuntos pendientes	No aplica	

Tabla 23:Gestionar fuentes.

Caso de Uso	Gestionar fuentes.
Objetivo	Gestionar las fuentes necesarias para el creador de paquetes.
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario del sistema accede a la opción “Fuentes”, el sistema permite administrar las fuentes del paquete de instalación, el caso de uso termina.
Complejidad	Alta
Prioridad	Alta

Precondiciones	-	
Postcondiciones	Se realiza la gestión de las fuentes,	
Flujo de eventos		
Flujo básico Gestionar dependencias		
	Actor	**Sistema**
1.	Selecciona la opción “AsiXMec”.	
2.		Muestra los datos a ser llenados: ✓ Oce Library. ✓ Fuentes. ✓ Main folder. ✓ Output ✓ Source ✓ Build form source ✓ Install depends automatically
3.	Selecciona el campo “Fuentes”.	
4.		Muestra una ventana emergente con las opciones que puede realizar: ✓ Agregar Fuentes (ver Sección 1). ✓ Modificar Fuentes (ver Sección 2). ✓ Eliminar Fuentes (ver Sección 3).
5.		Finaliza el caso de uso.
Sección 1: “Agregar Fuentes”		
Flujo básico		
	Actor	**Sistema**
1.	Selecciona el botón “Adicionar”.	
2.		Muestra una ventana emergente para poder seleccionar las Fuentes.
Flujos alternos		
2a. Cancelar la acción de agregar Fuentes.		
	Actor	**Sistema**
1.	Selecciona el botón “Cancelar”.	
2.		Vuelve a la pantalla de adicionar Fuentes.
Sección 2: “Modificar Fuentes”		
Flujo básico		
	Actor	**Sistema**
1.	Selecciona la Fuente a modificar y presiona el botón “Cambiar”.	
2.		Muestra una ventana emergente para poder seleccionar la Fuente que va a sustituir al anterior.
Flujos alternos		
2a. Cancelar la acción de modificar Fuentes.		
	Actor	**Sistema**
1.	Presiona el botón “Cancelar”.	

2.		Vuelve a la pantalla de adicionar Fuentes, con los Fuentes ya existentes.
Sección 3: “Eliminar Fuentes”		
Flujo básico		
1.	Selecciona el Fuentes a eliminar y presiona el botón “Eliminar”.	
2.		Muestra los Fuentes restantes en caso de existir.

Relaciones	**CU incluidos**	No aplica
	CU extendidos	No aplica
Requisitos no funcionales	No aplica	
Asuntos pendientes	No aplica	

ANEXO 2: Casos de prueba

Tabla 24. Descripción de valiables AsiXMec.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
V1	Seleccionar .pro	Cargar fichero	No	Permite cargar una ruta válida de un fichero y debe existir en la computadora.
V2	Directorio de compilación	Cargar directorio	No	Permite cargar una ruta válida de un directorio y debe existir en la computadora.
V3	Biblioteca OCE	Cargar directorio	No	Permite cargar una ruta válida de un directorio y debe existir en la computadora.
V4	Carpeta principal	Cargar directorio	No	Permite cargar una ruta válida de un fichero y debe existir en la computadora.
V5	Fuentes	Cargar fichero	No	Permite cargar una ruta válida de un fichero y debe existir en la computadora.

Tabla 25. Caso de prueba compilar.

Escenario	Descripción	V1	V2	Respuesta del sistema	Flujo central
EC 1.1 Introducir datos correctos en los campos.	La aplicación debe permitir la compilación del producto AsiXMec.	V /home/escritorio/ AsiXMec/ asixmec.pro	V /home/escritorio/ build	Se realiza la compilación del producto AsiXMec.	- Interfaz *AsiXMec*. - Introducir los datos.

					- Seleccionar el botón *Compilar*.

Tabla 26. Caso de prueba AsixMec.

Escenario	Descripción	V1	V2	V3	V4	V5	Respuesta del sistema	Flujo central
EC 1.1 Introducir datos correctos en los campos.	La aplicación debe permitir la creación del paquete.	V /home/escritorio/ AsiXMec/ asixmec.pro	V /home/escritorio/ build	V /opt/ oce	V /home/ escritorio/main	V /home/ Escritorio /fuentes/arial.ttf	Se crea el instalador de paquetes.	- Interfaz *AsiXMec*. - Introducir los datos. - Seleccionar el botón *Generar*.

Tabla 27. Descripción de las variables Sistema.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
V1	Aplicación de consola	checkbox	Si	Se puede seleccionar o no.
V2	Ejecutar como administrador	checkbox	Si	Se puede seleccionar o no.

Tabla 28. Caso de prueba Sistema.

Escenario	Descripción	V1	V2	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar la opción *Aplicación de consola*	La aplicación no se debe mostrar en el menú del sistema.	V seleccionado	V vacío	La aplicación no se muestra en el menú del	- Interfaz *Sistema*. - Seleccionar *Aplicación de*

				sistema.	*consola*. - Seleccionar el botón *Generar*.
EC 1.2. Seleccionar la opción *Ejecutar como administrador*	La aplicación debe necesitar de privilegios especiales y de contraseña *root.*	V	V	La aplicación solicita la contraseña *root.*	- Interfaz *Sistema*. - Seleccionar *Ejecutar como administrador*. - Seleccionar el botón *Generar*.
		vacío	seleccionado		
EC 1.3. Seleccionar ambas opciones.	La aplicación no se debe mostrar en el menú del sistema y se debe necesitar de privilegios especiales y de contraseña *root.*	V	V	La aplicación no se muestra en el menú del sistema y solicita la contraseña *root.*	- Interfaz *Sistema*. - Seleccionar *ambas opciones*. - Seleccionar el botón *Generar*.
		seleccionado	seleccionado		