UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 7

TÍTULO: *Diseñador de actualizaciones automáticas*

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

AUTORES: Beatriz Fernández Carmenate

Edison Garcia Puentes

TUTOR: Lic. Yasel Couce Sardiñas

Ciudad de la Habana, Junio 2010

"Año 52 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del presente trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los 24 días del mes de junio del año 2010.

Beatriz Fernández Carmenate

Firma del Autor

Edison Garcia Puentes

Firma del Autor

Lic. Yasel Couce Sardiñas

Firma del Tutor

AGRADECIMIENTOS

Beatriz:

A todos los que de una forma u otra han colaborado con mi formación profesional, especialmente a mis padres, a mi hermana y mi familia por el apoyo que siempre me han dado.
A mi novio Leonel por su apoyo y paciencia.
A Irina y Yaneysi mis amigas que han estado ahí siempre durante esta gran travesía.
A aquellos que directa o indirectamente me han ayudado en el desarrollo de mi tesis.
Al grupo de estudiantes y profesores del proyecto.
A todos los amigos, que me han ayudado a lo largo de la carrera.
A la dirección de la revolución por darme la oportunidad de estudiar en una universidad como esta.

Edison:

A mi tutor por su apoyo y confiar desde el primer momento en este éxito.
A todas aquellas personas que aportaron su granito de arena en la realización de este sueño.
A mi mamá y mi papá por ser siempre los que comprenden, los que apoyan, los que aconsejan, los que aman. Por ser los que siempre se quedan, los que nunca dan la espalda.
A mis abuelos Mima y Papi, por ser la segunda mayor influencia de mi vida. Son ustedes lo que amo en este mundo.
A mi hermanas Lisbet, Liset y Susel por escucharme, por quererme, por admirarme. Son la razón principal por la que quiero ser cada día mejor.
A mis tíos Jorge Luis, Lily y William, mis primos Raynel y Neysi por apoyarme en todo momento, brindarme grandes momentos y formar parte de mis mejores recuerdos.
A todas mis amistades, esas que son tantas y que por temor a pecar no podemos mencionar a algunos y a otros no.
A mi Cuqui y Yani por ser de las personas que llegan y nos marcan para siempre.
A todos los que fueron mis profesores por contribuir en mi formación.
A la Revolución, la FEU y todas aquellas personas vinculadas en mi vida como dirigente, gracias por darme el boleto a esa estación que nunca olvidaré.

DEDICATORIA

Beatriz

Le dedico la tesis a mis padres, que han sido los protagonista de mi día a día, y me han ayudado y apoyado para llegar hasta aquí.

A mi novio Leonel que me ayudó paso a paso en mi carrera para llegar hasta aquí, velando por mis buenos resultados y apoyándome para mejorar estos.

Edison

Dedico el presente trabajo de diploma:

A mi mamá por ser mi guía en estos años de universidad y dedicarme su vida entera.

A mis abuelos por ser mi inspiración diaria.

A mis hermanas por darme su apoyo siempre.

Este triunfo es para ustedes.

RESUMEN

El objetivo de la presente investigación es crear un sistema informatizado que permita gestionar el diseño y publicación de las actualizaciones automáticas para los productos que se desarrollan en el Centro de Informática Médica. Se diseñó una aplicación en ambiente de escritorio y se utilizó un repositorio, al cual se accede a través del protocolo FTP para publicar todos los recursos de actualización creados. Dicho repositorio organiza todas las actualizaciones por productos y estos por las distintas plataformas de sistemas operativos al cual pertenecen, de forma que se muestre más intuitiva y robusta dicha estructura.

El sistema obtenido está soportado sobre tecnología .NET, específicamente sobre la versión 2.0 de este marco de trabajo y como lenguaje de programación el C-Sharp. Para la codificación del mismo se utilizó como IDE de desarrollo el Visual Studio 2008. Se utilizó RUP, como metodología de desarrollo y CMMI como guía de proceso para la documentación de la aplicación creada.

La aplicación permite construir un perfil por producto con las principales características que describen las aplicaciones informáticas. Además brinda una gama de opciones para la construcción de actualizaciones compatibles con cualquier tipo de producto. La carencia de una solución de este tipo puede introducir errores en la creación de las actualizaciones, lo que se revierte finalmente en la desestabilización de las aplicaciones desplegadas por el Centro. El sistema obtenido en esta investigación constituye un impacto positivo sobre la estrategia de comercialización y despliegue de productos del Centro de Informática Médica.

ÍNDICE

INTRODUCCIÓN

La evolución de la informática y las telecomunicaciones en las últimas dos décadas ha colocado a los sistemas informáticos en un rol preponderante dentro de las instituciones. La misma dota a la humanidad de medios potentísimos para la expansión del conocimiento, la cultura y la sabiduría. Esta situación se ha visto potenciada por el auge de Internet, el desarrollo y ya habitual dependencia de los productos informáticos cada día más actualizados y con mayor calidad, donde los sistemas informáticos forman parte del corazón de la sociedad.

Cuba no está exenta dentro de esta gran gama de desarrollo científico y tecnológico. A pesar de verse frenada por el injusto bloqueo que le ha sido impuesto se han logrado avances incuestionables, comparables con países desarrollados. En diferentes sectores políticos, económicos y sociales el país ha obtenido buenos resultados. Poco a poco se han introducido en las diferentes empresas el uso de las computadoras, con fines económicos y en aras de un perfeccionamiento profesional.

Uno de los grandes proyectos de Cuba, es la creación de la Universidad de las Ciencias Informáticas (UCI). La UCI surge con la idea de vincular al estudiante en las tareas productivas referentes al desarrollo de software. Varias facultades crean software con diversos perfiles y logran que la UCI se inserte en el mercado internacional.

La facultad 7 de la UCI asume el reto de la informatización del Sistema de Salud Pública para la cual tiene una serie de productos de software correspondiente a esta línea. Estos productos informáticos crecen en funcionalidad progresivamente por la gran demanda de los mismos, debido al profundo proceso de transformación organizacional que manejan estas instituciones por el fulminante auge de la informática. Esto y el indiscutible hecho de que toda solución debe dar soporte y de que ningún software está libre de errores, crean la necesidad de definir un proceso para la actualización de los mismos.

Uno de los grandes problemas que se afrontan actualmente en la esfera del desarrollo de soluciones informáticas son las actualizaciones automáticas de las mismas. Desde hace varios años, este tema ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de productos informáticos. El mantenimiento del software es mucho más complejo que el mantenimiento del

hardware. Cuando un componente hardware se deteriora se sustituye por una pieza de repuesto, pero cada fallo o necesidad en el software implica un cambio importante en el sistema, solucionado por las actualizaciones automáticas.

Las actualizaciones automáticas son una parte importante de la seguridad global. Muchas formas de malware, especialmente los virus y gusanos, operan mediante la explotación de los defectos previamente inadvertido en los programas. El término explotar, cuando se utiliza como sustantivo en ciencias de la computación, se refiere a cualquier pieza de software que puede aprovechar alguna vulnerabilidad en un programa con el fin de obtener acceso no autorizado a cualquier programa.

Las actualizaciones ayudan a mantener el sistema actual con los parches de seguridad que reparan las fallas que los programas de malware intentan explotar. Solucionan, además, vulnerabilidades, problemas de seguridad o de funcionamiento del sistema, también se encargan de corregir errores, añadir nuevas funcionalidades y resolver problemas de estabilidad del software; por lo que se hace necesario tener el sistema totalmente actualizado.

En la actualidad diseñar manualmente un paquete de actualización, con las acciones y sus correspondientes recursos, traen consigo un gran esfuerzo y preparación. Esto constituye en el CESIM (Centro de Informática Médica) una tarea engorrosa por el conjunto de acciones que se deben realizar durante el empaquetado de la misma, además de las funcionalidades que debe permitir para gestionar actualizaciones de diferentes productos con particularidades diferentes.

El personal responsable de llevar a cabo el proceso de actualización debe tener presente las acciones que se realizaron en versiones anteriores y no perder la estructura del árbol de directorios. Para de esta forma poder detallar manualmente las acciones de una nueva versión de software, bastante costoso en tiempo.

Lo anterior planteado, no asegura que estas personas construyan el paquete de actualización de forma detallada y libre de errores, por lo que este proceso de forma manual traería una serie de problemas que surgen por:

- La carencia de un historial de versiones: Productos con versiones inferiores a la última actualización generada, no son capaces de actualizarse por incompatibilidades generadas por la pérdida de información.
- Se ignora en muchos casos la versión en la que se está trabajando y los cambios que se han hecho hasta el momento. Sin estos elementos no hay forma de recuperarse de una situación que pueda presentarse durante el proceso de actualización.
- No está definida una estructura de forma general en el repositorio donde se publiquen todas las versiones de cada producto elaborado en el CESIM.

Dada la situación anterior, el **problema** a resolver consiste en la carencia de una solución informática que permita la gestión de las actualizaciones automáticas para las soluciones desarrolladas en el Centro de Informática Médica.

El **objeto de estudio** se centra en el proceso de gestión de la información de las aplicaciones informáticas desarrolladas en el Centro de Informática Médica y el **campo de acción** se enmarca en el proceso de gestión de la información de las actualizaciones automáticas para las soluciones elaboradas en el Centro de Informática Médica.

Para dar solución al problema antes mencionado se propone como **objetivo general**: Implementar una aplicación informática para la gestión de actualizaciones automáticas, de manera que facilite el proceso de actualización de software elaborados en el Centro de Informática Médica.

Para cumplir con el objetivo anteriormente planteado y resolver la problemática que ocupa esta investigación se proponen las siguientes tareas:

- Identificar las principales funcionalidades que componen un diseñador de actualizaciones para garantizar un producto acorde con las tendencias actuales.
- Proponer una estructura única para las actualizaciones automáticas, facilitando la ubicación organizada de la versión de un producto.
- Seleccionar la tecnología sobre la cual se implementará el diseñador.
- Diseñar la propuesta de solución, de manera que se garantice el uso de buenas prácticas en la implementación de la aplicación.

- Implementar el diseñador de actualizaciones automáticas de manera que automatice el diseño y publicación de actualizaciones.

El presente trabajo cuenta de cuatro capítulos estructurados de la siguiente forma:

Capítulo 1 Fundamentación Teórica: Aborda el estado del arte del tema a tratar a nivel internacional, nacional y de la universidad e incluye una explicación de las técnicas, tecnologías, metodologías y software empleados en la investigación para darle solución al problema. Se tratan los conceptos fundamentales para una correcta comprensión del tema.

Capítulo 2 Características del sistema: Define las características del sistema y se enmarca en el objeto de estudio. Se hace un análisis del dominio de la aplicación, se describen los procesos a automatizar para darle solución al problema y se generan documentos referentes a esta fase. También se definen requisitos no funcionales y un prototipo de interfaz externa.

Capítulo 3 Diseño del sistema: Primeramente se realiza el diseño del sistema donde se define el diagrama de clases del diseño por casos de uso así como la relación existente entre ellas. Posterior a esto se muestra la interacción entre los actores y el sistema mediante los diagramas de secuencia. Además se especifica la seguridad y el diseño de la interfaz de la aplicación.

Capítulo 4 Implementación: Presenta el modelo de implementación del Diseñador de actualizaciones automáticas, compuesto por el diagrama de componentes y el modelo de despliegue.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se hace mención a un grupo de conceptos que son importantes para el desarrollo de la investigación y que son utilizados en el resto del documento. Asimismo se efectúa un análisis del estado de los sistemas de actualizaciones automáticas a nivel nacional y en el ámbito internacional. Además se expone las principales características de las herramientas, tecnologías y la metodología empleadas en el desarrollo del sistema.

1.1 Conceptos asociados al dominio del problema

Para mejor entendimiento y desenvolvimiento de las áreas temáticas que se abordarán en el presente y posteriores capítulos, se mostrarán una serie de conceptos identificados durante la investigación realizada.

En informática, una ***aplicación*** es un programa informático diseñado para facilitar al usuario la realización de un determinado tipo de trabajo. Suele resultar una solución informática para la automatización de ciertas tareas complicadas como pueden ser la contabilidad o la gestión de un almacén. Ciertas aplicaciones desarrolladas “a medida” suelen ofrecer una gran potencia ya que están exclusivamente diseñadas para resolver un problema específico. [1]

Se designa con el término ***actualización informática*** a aquella tarea o actividad que supone la puesta al día y/o sustitución de una información contenida en un registro o archivo por otra más reciente. [2]

Las ***actualizaciones automáticas*** permiten mantener el software actualizado de forma automática. Las actualizaciones de software incluyen revisiones, revisiones rápidas y paquetes de servicio. Un paquete de servicio es una actualización periódica que corrige los problemas con una versión en particular de un producto. Las revisiones y revisiones rápidas son actualizaciones que se liberan rápidamente para corregir problemas de seguridad y estabilidad con un sistema o un producto de software. [3]

1.2 Antecedentes del sistema

Un sistema de software instalado está en constante evolución a través de extensiones, mantenimiento, cambios en los requerimientos o cambios en la configuración. Manejar la evolución de soluciones informáticas liberadas e instaladas es un problema complejo y frecuentemente subestimado que puede ser causa de dificultades.

El proceso de actualización de software puede ser visto como el movimiento de una configuración a otra por adición, eliminación, reemplazo o reconfiguración de las funcionalidades del software. [4]

En la actualidad se han creado herramientas con el propósito de cubrir las actividades relacionadas con la actualización del software. La creación de un paquete de actualización y su publicación dio paso a la creación de sistemas de gestión de paquetes con el fin de mantener al cliente sus productos actualizados.

Un **sistema de gestión de paquetes**, también conocido como gestor de paquetes, es una colección de herramientas que sirven para automatizar el proceso de instalación, actualización, configuración y eliminación de paquetes de software. El término se usa comúnmente para referirse a los gestores de paquetes en sistemas Unix-like, especialmente Linux, ya que se apoyan considerablemente en estos sistemas de gestión de paquetes.

En estos sistemas, el software se distribuye en forma de paquetes, frecuentemente encapsulado en un solo fichero. Estos paquetes incluyen otra información importante, además del software mismo, como pueden ser el nombre completo, una descripción de su funcionalidad, el número de versión, el distribuidor del software, la suma de verificación y una lista de otros paquetes requeridos para el correcto funcionamiento del software. Esta meta información se introduce normalmente en una base de datos de paquetes local.

Normalmente se pone a disposición de los usuarios en los repositorios, con el fin de proporcionar a los usuarios de un sencillo control sobre los diferentes tipos de software que van a instalar en su sistema y, en ocasiones, debido a razones legales o conveniencias por parte de los distribuidores.

Los sistemas de gestión de paquetes tienen la tarea de organizar todos los paquetes instalados en el sistema y se encargan de mantener su usabilidad. Esto se consigue combinando las siguientes técnicas:

- Comprobación de la suma de verificación para evitar que haya diferencias entre la versión local de un paquete y la versión oficial.
- Instalación, actualización y eliminación simple de paquetes.
- Resolución de dependencias para garantizar que el software funcione correctamente.
- Búsqueda de actualizaciones para proveer la última versión de un paquete, ya que normalmente solucionan bugs y proporcionan actualizaciones de seguridad.
- Agrupamiento de paquetes según su función para evitar la confusión al instalarlos o mantenerlos.

1.2.1 Sistemas existentes en el ámbito internacional

Por la naturaleza del software libre, los paquetes bajo licencias compatibles y similares están disponibles para usarlo en varios sistemas operativos. Estos paquetes pueden ser fácilmente combinados y distribuidos usando "packaging systems" configurables para manejar los diferentes cambios del software y administrar las dependencias y los conflictos específicos de una versión. Algunos "packaging systems" de software libre son ellos mismos liberados como software libre.

Advanced Packaging Tool (Herramienta Avanzada de Empaquetado), abreviado APT, es un sistema de gestión de paquetes creado por el proyecto Debian. APT simplifica en gran medida la instalación y eliminación de programas en los sistemas GNU/Linux.

No existe un programa apt en sí mismo, sino que APT es una biblioteca de funciones C++ que se emplea por varios programas de línea de comandos para distribuir paquetes. En especial, apt-get y apt-cache. Existe un repositorio central con más de aproximadamente 25.000 paquetes apt utilizados por apt-get y programas derivados para descargar e instalar aplicaciones directamente desde Internet, conocida como una de las mejores cualidades de Debian.

Advanced Packaging Tool (APT)	
Desarrollador:	Proyecto Debian
Escrito en:	C
Sistema Operativo:	GNU/Linux, OpenSolaris y Mac OSX
Género:	Sistema de Gestión de Paquetes

Licencia:	GNU GPL

Yellow dog Updater, Modified (YUM) es una herramienta libre de gestión de paquetes para sistemas Linux basados en RPM. Fue desarrollado por Seth Vidal y un grupo de programadores voluntarios, y actualmente se mantiene como parte del proyecto Linux@DUKE de la Universidad de Duke.

Yum fue desarrollada principalmente para actualizar y controlar los sistemas Red Hat utilizados en el departamento de física de la Universidad de Duke. Desde entonces, ha sido adoptada por Fedora, CentOS, y otras distribuciones de GNU/Linux basadas en RPM, incluyendo el mismo Yellow Dog, donde reemplazó a la utilidad original YUP. El manejador de paquetes de Red Hat, up2date, también puede hacer uso de los repositorio de software de yum cuando realiza actualizaciones de software. Red Hat Enterprise 5 reemplazó up2date por yum y pirut.

Con los paquetes "yum-updatesd" o "yum-updateonboot" se puede hacer una actualización de software automática.

El sistema de repositorios yum está convirtiéndose rápidamente en un estándar para los repositorios basados en RPM. En SUSE Linux 10.1 se añade soporte para repositorios YUM en YaST, y los repositorios de openSUSE están basados exclusivamente en Yum.

Yellow dog Updater, Modified está disponible bajo licencia GNU GPL versión 2 o superiores

Advanced Packaging Tool (APT)	
Desarrollador:	Seth Vidal
Sistema Operativo:	Linux
Género:	Sistema de Gestión de Paquetes
Licencia:	GNU GPL

Sistemas propietarios

En la actualizad existen sistemas operativos propietarios que usan sistemas de gestión de paquetes:

- El comando de AIX para las bases de datos Object Data Manager (ODM) es installp.
- El formato SysV usado por Solaris.
- Software Distributor es el gestor de paquetes de HP-UX.
- En el framework .NET de Microsoft, un ensamblado es una biblioteca de código parcialmente compilado destinado al uso en deployment, versioning y seguridad. [5]

Updater Application Block (UAB)

El bloque del actualizador, creado por Microsoft, es una biblioteca que puede agregar a su aplicación para administrar la descarga de las partes de la aplicación a través de HTTP. Las actualizaciones se procesan y se muestran todos los archivos de la nueva versión de la aplicación en un manifiesto. La aplicación debe cambiar de forma considerable para poderla utilizar.

UAB se trata claramente de una medida temporal mientras Microsoft desarrolla una solución definitiva. Esta solución es ClickOnce. [6]

ClickOnce

Es una tecnología de despliegue de aplicaciones windows forms (principalmente Smart Clients), basada en el concepto de publicación en un servidor Web, FTP o de archivos compartidos, para permitir la descarga, instalación y actualización automática de la aplicación.

El core de esta tecnología de despliegue está basado en dos archivos XML: un archivo de manifiesto de aplicación y un archivo de manifiesto de despliegue.

- ✓ Manifest file: El manifiesto de aplicación (.manifest file) describe los ensamblados y archivos que comprenden la aplicación, incluyendo información de identidad de los ensamblados (nombre, hash/public key token, versión, y localización), dependencia de la aplicación, e información de confianza que afecte las políticas de seguridad bajo la cual la aplicación está ejecutándose.

- ✓ Application file: El manifiesto de despliegue (.application file), contiene un resumen de la información de despliegue en sí misma. incluyendo información de versión, donde encontrar el

manifiesto de la información, y el número de opciones respecto al comportamiento en la clase de despliegue y actualización que se llevará cabo. [7]

Entre las ventajas que ofrece ClickOnce:

- ✓ Las actualizaciones en ClickOnce son transaccionales, es decir, se asegura que se realice completa y satisfactoriamente, y si esto no es posible, entonces no se realiza la actualización.
- ✓ Los manifiestos de aplicación e instalación que genera ClickOnce están firmados digitalmente, esto proporciona la seguridad de que la aplicación y sus actualizaciones provengan de una fuente segura y fiable.
- ✓ Con ClickOnce tienes la opción de volver a la versión anterior de la aplicación (*rollback version*) desde el servidor (reemplazando el manifiesto de instalación) o desde el cliente (en Añadir o Remover programas).
- ✓ ClickOnce optimiza el proceso de actualización bajando sólo los ficheros que han cambiado, para esto, asocia cada fichero con una clave única para saber si ha cambiado y también para verificar si un archivo ha sido manipulado (*file tampering*). [8]

Desventajas que tiene ClickOnce:

- ✓ Al publicar una actualización para la aplicación esta no "sobre escribe" la versión anterior de la aplicación. Por lo tanto los archivos xml que tenían los usuarios antes de la actualización no se toman en cuenta y prácticamente el usuario "pierde" la configuración que tenía y debe volver a configurar.
- ✓ Un problema común con las aplicaciones que utilizan ClickOnce como mecanismo de distribución es que los proxy con autenticación no permiten la descarga de los componentes de la aplicación.
- ✓ Es propietario y cuesta su adquisición.
- ✓ Están completamente auto-contenidas e instaladas por usuario, lo que implica que no son necesarios derechos de administración. La ruta donde se instala ClickOnce es: [9]
    - o WindowsVista: C:\Users\[nobreusuario]\AppData\Local\Apps\2.0\
    - o WindowsXP: E:\Documents and Settings\[nobreusuario]\AppData\Local\Apps\2.0\

Cada aplicación ClickOnce que instale en el equipo obtiene su propio directorio de datos, el cuál se almacena en la carpeta *Documents and Settings*. Todo archivo incluido en una aplicación ClickOnce y marcado como archivo de "datos" se copia en este directorio al instalar la aplicación.

ClickOnce trae por defecto esta ruta de instalación y cuando se realiza el cambio a una ruta de conveniencia muchos de sus componentes dejan de funcionar. Asimismo, si se mantiene esta ruta trae como resultado que todos los usuarios cuando descarguen una versión de actualización de un software, esta se encuentre repetida en cada sección de usuario. [10]

Updater Studio

Solución .NET que se realizó con el objetivo de crear y administrar las actualizaciones. Con esta solución se gestiona todo el proceso de generación de archivos de actualización y asignarle acciones a los mismos. Todos los archivos son cargados automáticamente a su servidor al concluir el proceso de asignación de responsabilidades a los mismos.

La versión express es gratuita.

1.2.2 Sistemas existentes en la UCI

La Universidad de las Ciencias Informáticas (UCI) desarrolla en estos momentos un conjunto de proyectos que manejan ya la solución de los problemas de las actualizaciones automáticas de los mismos así estos brindan un mejor soporte y seguimiento de estos. Entre los que se abarcan la automatización de los Registros de Notarias, la Identificación de los Ciudadanos y con el objetivo de automatizar los procesos relacionados con la identificación, migración y extranjería en la República Bolivariana de Venezuela, surge el sistema SAIME (Servicio Autónomo de Identificación Migración y Extranjería) también de este proyecto.

Con el objetivo de mejorar el proceso de actualización del sistema SAIME, el proyecto Identidad crea un sistema que se basa en el mecanismo de actualización Updater Application Block (UAB). Su sistema incluiría una serie de mejoras que vendrían dadas por las modificaciones y re-implementación del

mecanismo de actualización que se encontraban usando y le agregarían además algunas funcionalidades que no estaban desarrolladas. Para la publicación de la nueva versión, cuentan con un servidor central donde publican los recursos de actualización además de la documentación o los archivos ".docum" que contienen la información necesaria para realizar el proceso de actualización en los clientes. Así fue como surgió el sistema de Identidad que actualmente se encuentra en Venezuela facilitando servicios de actualizaciones para sus aplicaciones.

El proyecto Registro de Notarías también tiene gran parte de su funcionalidad implementada usando la Plataforma .Net, por lo que se ajustó el mecanismo construido por el proyecto Identidad a fin de automatizar el proceso.

En el departamento de Software Médico Imagenológico (SWMI) también existe una solución para actualizar sus productos de software en la República Bolivariana de Venezuela y lo hacen llamar "ClickOnce cubano".

El "ClickOnce cubano" es una herramienta que se basa en la tecnología de actualización ClickOnce de la plataforma .NET, la misma utiliza dos componentes un updater y un publisher. Con el publisher publican en un servidor central las versiones de un software determinado y el updater es el encargado de actualizar el nodo cliente a través de la lectura de un documento XML que se encuentra en el servidor central. Esta herramienta implementa las funcionalidades principales de ClickOnce. La desventaja que tiene es que actualiza todos los componentes de una nueva versión del sistema aunque solo un componente de esta versión sea diferente al de la versión anterior. No desarrollan la dependencia entre elementos y aún no incorporan ficheros de reglas junto a los manifiestos (documentos XML que describen las actualizaciones) que ya poseen.

1.2.3 Análisis comparativo con la propuesta de solución

Las herramientas descritas anteriormente automatizan muchos de los procesos que se necesitan para realizar una actualización automática.

ClickOnce es una herramienta muy potente pero posee varias desventajas que hicieron que el departamento de Software Médico Imagenológico (SWMI) se diera a la tarea de crear su propio

actualizador, que a pesar de resolver algunos de los problemas de ClickOnce este aún tiene características que lo superan en cuanto a la actualización de los componentes que requieren ser actualizados y que para escenarios en los cuales no desea que los usuarios tomen sus propias decisiones de confianza, puede personalizar el comportamiento mediante la firma de los manifiestos; aún así, es débil en cuanto al proceso de actualización de productos de software grandes.

Las aplicaciones que requieren tareas adicionales, como modificar el registro, no son compatibles con la tecnología de implementación de ClickOnce, cuando con esta herramienta se hace alguna modificación a algún programa y compilas, este automáticamente duplica esa actualización, esto no se ajusta a lo que se quiere desarrollar porque se estaría atado a la tecnología visual estudio y se quiere poder actualizar cualquier tipo de aplicación.

Cuando se trata de realizar una actualización en Linux cada vez que hay una nueva versión él te compila un binario y lo que hace es sobre escribir la instalación que se tiene actualmente, este binario se publica en el repositorio como un binario completo entonces en el cliente se tiene que desinstalar la aplicación y volver a instalar, esto no es lo que se busca con la aplicación, se quiere poder publicar pequeños cambios en un único paquete. No se quiere estar atado a ningún IDE ni a cambios muy bruscos en las aplicaciones.

El proyecto Identidad se basó durante su investigación en el Updater Application Block (herramienta que le dio paso a la creación de ClickOne). Su aplicación posee características muy buenas pero fue desarrollada para SAIME y posee características que no cumplen todas las necesidades que se necesitan para el desarrollo del paquete de actualización de las aplicaciones del CESIM de la UCI. Se desea configurar tareas asociar archivos a estas tareas para la creación de una versión y con esto resolver las situaciones planteadas.

Después de analizar profundamente todas las herramientas empleadas a nivel internacional y en nuestra Universidad, para darle solución al problema que se plantea se propone una aplicación de escritorio que permita a un representante de un proyecto del CESIM de la UCI gestionar todo el proceso de creación de una actualización. Cada actualización de un producto de software se va a guardar en un nodo central

donde se van a publicar todas las versiones de cada una de estas aplicaciones, de forma tal que cuando un cliente solicite una actualización del software que posee pueda obtener su última versión.

Para su realización y desarrollo se hace necesario realizar un estudio de las tecnologías, lenguajes y metodologías óptimas.

1.3 Tecnologías, lenguajes, metodologías utilizadas y protocolos.

1.3.1 Lenguaje de programación.

En el mundo del desarrollo del software muchos programadores usan lenguajes de programación de alto nivel y orientados a objetos; la decisión del uso de uno u otro para el desarrollo de un software determinado está en las librerías que estos utilizan, en las cuales radica la verdadera riqueza del lenguaje. La elección final del lenguaje también dependerá del posible conocimiento que se tenga de la sintaxis del mismo y de su adaptación al medio para el que se quiere programar, ya que es diferente programar para red local, para Windows, Mac o Linux. A continuación se presentarán algunas características, ventajas y desventajas del lenguaje de programación seleccionado para trabajar.

1.3.1.1 El lenguaje C#

Actualmente se está utilizando con gran efectividad el nuevo lenguaje C# desarrollado por la empresa Microsoft Corporation, como una recopilación de lo mejor de C++ y Java. Tiene algunas ventajas sobre los restantes lenguajes de alto nivel orientados a objetos. El estándar del lenguaje C# por excelencia está comprendido en la especificación ECMA-334 de la Ecma International.

C# es un lenguaje orientado a objetos, con seguridad de tipos que permite a los desarrolladores crear una amplia gama de aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Puede utilizar este lenguaje para crear aplicaciones cliente para Windows tradicionales, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, y otras tareas.

C# admite los conceptos de encapsulación, herencia y polimorfismo. Una clase puede heredar directamente de una clase primaria, pero puede implementar cualquier número de interfaces. En C#, una estructura es como una clase sencilla; es un tipo asignado en la pila que puede implementar interfaces

pero que no admite la herencia. El proceso de generación de C# es simple en comparación con el de C y C++, y es más flexible que en Java. No hay archivos de encabezado independientes, ni se requiere que los métodos y los tipos se declaren en un orden determinado. Un archivo de código fuente de C# puede definir cualquier número de clases, estructuras, interfaces y eventos. [11]

Ventajas:

- ✓ Compila a código intermedio independiente del lenguaje en que haya sido escrita la aplicación e independiente de la máquina donde vaya a ejecutarse.
- ✓ Realiza la recolección automática de basura.
- ✓ Posee capacidades de reflexión.
- ✓ Es flexible en cuanto al orden de definición de las clases y las funciones.
- ✓ Soporta definición de clases dentro de otras.
- ✓ Todos los valores son inicializados antes de ser usados (automáticamente por defecto, o manualmente desde constructores estáticos).

1.3.2 Plataforma .NET

En 1998 un equipo de trabajo de Microsoft Corporation comenzó a trabajar en el proyecto Próxima Generación de Servicios Windows (en Inglés *Next Generation Windows Services*) el cual se fusionó con el grupo encargado de liberar Visual Studio 7 con el objetivo de desarrollar un entorno común de ejecución para todos los lenguajes cubiertos por esta herramienta de desarrollo que permitiera a terceras empresas crear lenguajes adaptados al entorno. Luego, en el año 2000 Microsoft publicó este trabajo denominado Microsoft.NET.

La plataforma .NET sirve de mediador entre el programador y las particularidades del sistema operativo para el que se programen las aplicaciones. Una vez terminado el programa, su ejecución se realizaría sobre esta plataforma que entonces mediaría entre él mismo y el sistema operativo. De esta forma, un sistema desarrollado para .NET pudiera ejecutarse en cualquier sistema operativo que tenga instalada una versión de este *framework* como también se le denomina.

Está diseñado para utilizar los servicios web con XML como mecanismo principal de comunicación entre aplicaciones. Posee avanzadas funciones en tiempo de ejecución lo que permite que cualquier aplicación pueda ser convertida en servicios web XML. Permite escribir programas en cualquiera de los lenguajes soportados por la plataforma, incluso utilizar simultáneamente varios lenguajes en un mismo programa. Entre los lenguajes que se han adherido a esta familia se encuentra el ya mencionado C#. Agrupa además al Microsoft Visual Basic, C++, Java, Pascal, entre otros conformando un grupo de más de veinte. Para su óptimo aprovechamiento se desarrolló Visual Studio 2005 que ha evolucionado hasta las actuales versiones de Visual Studio 2008.

Acciones como conexiones a bases de datos y la creación de componentes visuales se encuentran empaquetadas en componentes que tienen implementadas todas las funciones necesarias para estos propósitos. Solo basta con arrastrarlos hacia la aplicación y utilizar sus ventajas. Realiza una adecuada gestión de memoria haciendo las aplicaciones más confiables. Utiliza la ejecución en paralelo aportando mayor eficiencia en tiempo de ejecución.

Tiene un componente de seguridad capaz de monitorear las acciones que pueden ser sensibles sobre el sistema operativo controlando quién escribe y ejecuta el código y con qué propósitos lo hace. Inicialmente el framework no era gratuito, más tarde Microsoft liberó la versión 2.0 y actualmente se encuentra disponible para su descarga desde el portal web de esa empresa. [12]

1.3.3 Framework 2.0

Microsoft .NET Framework versión 2.0 Redistributable Package instala el entorno en tiempo de ejecución y los archivos asociados de .NET Framework necesarios para ejecutar aplicaciones desarrolladas para .NET Framework v2.0.

.NET Framework versión 2.0 mejora la escalabilidad y el rendimiento de aplicaciones gracias a características mejoradas como el almacenamiento en caché, el desarrollo de aplicaciones y además, es compatible con la gama más amplia de exploradores y dispositivos con servicios y controles ASP.NET 2.0.

El .NET Framework fue diseñado para cumplir con objetivos como: proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de

forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota; proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones, la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza, que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos; basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código. .NET tiene dos componentes principales: *Common Language Runtime* (CLR) y la biblioteca de clases de .NET Framework (BCL - *Basic Class Library*). [13]

1.3.4 Selección de un lenguaje de programación

Este lenguaje ha alcanzado gran auge por su sencillez y se utiliza para desarrollar gran cantidad de aplicaciones desde mediano a gran tamaño, obviando esfuerzo de programadores en actividades de rutina. Está debidamente estandarizado. Se usa en múltiples grupos de proyecto en la Facultad 7 de la UCI. Es gratis tanto el C# como la plataforma que lo soporta. Resulta, por tanto, el adecuado para las aspiraciones y necesidades de este trabajo.

1.3.5 XML (*Extensible Markup Language*).

Es un lenguaje de etiquetado extensible muy simple, pero estricto, que juega un papel fundamental en el intercambio de una gran variedad de datos. Sirve para estructurar, almacenar e intercambiar información. Es la base de los servicios Web, el contenido almacenado en un documento XML se puede transferir fácilmente a través de la red. Los servicios Web XML actúan de forma independiente y además permiten que las aplicaciones compartan información e invoquen funciones de otras aplicaciones independientemente del sistema operativo o la plataforma en que se ejecutan y los dispositivos utilizados para obtener acceso a ellos. [14]

1.3.6 XSchema XML

XSchema es un lenguaje de definición de clases de documentos XML. Realmente un documento XSchema es un documento XML con unas marcas determinadas. A los documentos XML derivados de un documento XSchema se les suele denominar “documentos instancia”.

XSchema dispone de una amplia gama de tipos básicos predefinidos para la definición de atributos y elementos. Estos tipos básicos están compuestos en su mayoría por tipos simples, aunque también podemos encontrar un tipo lista y un tipo unión. También podemos construir nuestros propios tipos de datos, extendiéndolos de los existentes o creando estructuras nuevas.

XSchema es un lenguaje muy potente para definir documentos XML y nos será de gran ayuda para validar los documentos XML que generemos o utilicemos.

1.3.7 Modelo de Desarrollo del Software (CMMI).

CMMI (Modelo Integrado de Madurez de Capacidades) no es más que una guía para mejorar procesos y comprobar la capacidad de un grupo al ejecutarlos, un modelo de madurez, un marco para diagnosticar el estado de la mejora que indica que deben hacer los procesos. Este contiene dos representaciones: la continua y la escalonada. La continua presenta 6 niveles de proceso y la escalonada tiene cinco niveles de madurez. La utilizada para este trabajo es la escalonada y el nivel 2 de madurez de CMMI.

Esta representación ofrece una manera estructurada y sistemática de enfrentar la mejora de proceso, ejecutando un paso cada vez. El nivel 2 de madurez de CMMI se define en siete áreas de control que son REQM (Gestión de Requerimientos), PP (Panificación de Proyecto), PMC (Seguimiento y Control de Proyectos), SAM (Gestión de Acuerdos con Proveedores), MA (Medición y Análisis), PPQA (Aseguramiento de la Calidad de Procesos y Productos), CM (Gestión de la Configuración). El área de control que se utilizara es REQM, el objetivo de esta función es administrar los requisitos de los productos y componentes de productos del proyecto. Identificar inconsistencias entre esos requisitos y los planes y productos de trabajo del proyecto. Como metodología de desarrollo se utilizará RUP, establecida por la UCI como metodología a utilizar con dicho modelo.

Un Proceso de Desarrollo de Software es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto. Constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos y junto con el Lenguaje Unificado de Modelado UML 2.0.

El Lenguaje de Modelado Unificado (UML *Unified Modeling Language*) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de actividades concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

1.3.8 Arquitectura del Sistema.

La arquitectura en capas es la generalización de la arquitectura Cliente-Servidor, donde la carga se divide en dos partes con un reparto claro de funciones: una Capa de Presentación y otra parte para las reglas lógicas del negocio, denominada Capa de Negocio, pudiéndose crear nuevas capas intermedias de ser necesario. Es decir, está soportado sobre un nivel de abstracción creciente, lo cual permite a los desarrolladores la fragmentación de un problema complejo en una secuencia de pasos incrementales. También proporciona una amplia reutilización, pues los datos abstractos pueden ser utilizados por diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces. [15]

La utilización de este patrón quiere decir que se mantendrá una clara separación entre la lógica de negocio y la presentación. Permitiendo flexibilidad y facilidad a la hora de realizar posibles modificaciones.

1.3.9 Herramientas utilizadas.

El actual desarrollo de las tecnologías y conjunto a eso el desarrollo de las aplicaciones libres (Software Libre) como el comienzo de una nueva etapa en el desarrollo de la Informática y las Comunicaciones. La implementación de todos los componentes y del servicio de actualización se realizó sobre la tecnología .Net de Microsoft, con el Microsoft .Net Framework y como entorno de desarrollo, el Microsoft Visual Studio .Net 2008. Se debe destacar que al no existir otra tecnología implicada, además del .Net Framework de Microsoft, también es posible usar estos componentes e incluso desarrollar extensiones para el proceso de actualización.

1.3.9.1 Visual Studio 2008

Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET. Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. [16]

1.3.9.2 Enterprise Architect 7.1:

Enterprise Architect (EA) es una herramienta CASE (Computer Aided Software Engineering) para el diseño y construcción de sistemas de software. EA soporta la especificación de UML 2.0, que describe un lenguaje visual por el cual se pueden definir mapas o modelos de un proyecto.

EA es una herramienta progresiva que cubre todos los aspectos del ciclo de desarrollo, proporcionando una trazabilidad completa desde la fase inicial del diseño a través del despliegue y mantenimiento. También provee soporte para pruebas, mantenimiento y control de cambio.

Algunas de las características claves de Enterprise Architect son:

- ✓ Generar código para el software que está construyendo.
- ✓ Crear elementos del modelo UML para un amplio alcance de objetivos.
- ✓ Ubicar esos elementos en diagramas y paquetes.
- ✓ Crear conectores entre elementos.
- ✓ Documentar los elementos que ha creado.
- ✓ Realizar ingeniería reversa del código existente en varios lenguajes.
- ✓ Importación/Exportación XMI 2.1.

Modelado basado en el Equipo

- ✓ Archivos compartibles o modelos basados del repositorio.

- ✓ Control de versiones con cualquier herramienta SCC.
- ✓ Seguridad incorporada y administración de permisos.

Usando EA, puede realizar ingeniería directa y reversa de código C++, C#, Delphi, Java, Python, PHP, VB.NET y clases de Visual Basic, sincronizar códigos y elementos del modelo, diseñar y generar elementos de base de datos. La documentación de alta calidad puede ser rápidamente exportada desde sus modelos en industria estándar .formato RTF e importar a Word para una personalización y presentación final.

Enterprise Architect sustenta todos los diagramas y modelos UML. Puede modelar procesos de negocio, sitios web, interfaces de usuario, redes, configuraciones de hardware, mensajes y más. Estimar el tamaño de su proyecto en esfuerzo de trabajo en horas. Capturar y trazar requisitos, recursos, planes de prueba, solicitudes de cambio y defectos. Desde los conceptos iníciales hasta el mantenimiento y soporte, Enterprise Architect tiene las características que precisa para diseñar y administrar su desarrollo e implementación. [17]

En este capítulo se realizó un análisis de las ventajas de desarrollar un buen proceso de actualización de un software y se profundizó en las características de los sistemas informáticos que se utilizan a nivel internacional y nacional para este proceso. Se seleccionó dentro de un grupo de tecnologías, herramientas y metodologías las adecuadas para realizar el software que se requiere: C# como lenguaje de programación, Framework .NET de Microsoft, Enterprise Architect como herramienta CASE y RUP como metodología de desarrollo.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

El presente capítulo tiene como objetivo la descripción de las características del sistema además de la propuesta de solución para el Diseñador de Actualizaciones Automáticas. Debido a la poca claridad de los procesos del negocio se desarrolla un Modelo de dominio, además se analizan los requerimientos funcionales y no funcionales lo cual permite el mejor entendimiento del sistema.

2.1. Objetos de automatización

Los problemas que implica el desempeño manual de las actividades de actualización de los proyectos que se desarrollan en el CESIM y la importancia que tiene el soporte y el mantenimiento de estos, crean la necesidad de automatizar dichas actividades a través de un mecanismo de actualización.

Para diseñar una actualización se debe tener en cuenta toda la información que pueda brindar el perfil de un producto y además las acciones que se deben llevar a cabo para completar una actualización. Durante el estudio realizado en el capítulo anterior existían herramientas como el Updater Studio que manejaba lo referente a varios requerimientos de rendimiento de hardware para el correcto funcionamiento de una nueva versión. Con esto, en caso de no cumplir con estos requerimientos se le informaba al cliente del poco rendimiento que podía tener la nueva versión de la aplicación si se obviaba ese paso, permitiendo continuar con la instalación del release.

Esto permitió enriquecer los elementos del perfil del producto, que completarían junto con la asignación de responsabilidades a ficheros, todo el proceso del diseño de la actualización que se desea automatizar. Todas las operaciones referentes a las actualizaciones son llevadas a cabo por el proveedor de actualizaciones.

2.2. Información que se maneja

La información utilizada en los procesos a automatizar las actualizaciones del software son varias, estas describen no solo la información, sino también el formato.

- ✓ Listado de los recursos necesarios: Fichero XML que contiene el listado de la información de todos los recursos que el Actualizador necesita para actualizar la aplicación con todas las acciones que este debe realizar.
- ✓ Paquete de actualización: Son todos los recursos cuya descripción está contenida en el "Listado de recursos necesarios" empaquetados y compactados en un fichero ZIP.
- ✓ Perfil del producto: Describe la información del conjunto acciones que se puedan realizar sobre los recursos.
- ✓ Estructura de aplicación: Árbol de directorios que conforma la aplicación después de instalada y que puede sufrir cambio con la aparición de una nueva versión.

2.3. Descripción del Sistema

El Diseñador se ha dividido las funcionalidades informáticas en dos partes importantes con responsabilidades definidas de la siguiente forma:

- ✓ La funcionalidad "Diseñador de Información" es el encargado de diseñar las actualizaciones de forma automática, permitiendo visualizar el estado de dicho proceso.
- ✓ La funcionalidad "Publicador de Información" es la encargada de la interacción con el responsable de la actualización, permitiéndole visualizar el estado del proceso general a través de una interfaz y de mantener publicadas las nuevas versiones existentes de los productos.

2.4. Modelo de Dominio

El modelo de dominio 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. Representa conceptos del mundo real, no de los componentes de software. Una clase conceptual puede ser una idea o un objeto físico (símbolo, definición y extensión).

El modelo de dominio se representa en UML con un Diagrama de Clases en los que se muestra:

- ✓ Conceptos u objetos del dominio del problema: clases conceptuales.
- ✓ Asociaciones entre las clases conceptuales.

- ✓ Atributos de la clase conceptuales.

En el Modelo de Dominio no se muestra comportamiento. Las clases conceptuales pueden tener atributos pero no métodos. Cualquiera sea la solución de casos de uso que se haya elegido, los conceptos e ideas propias del dominio del problema son las mismas; un mismo modelo de dominio contempla cualquiera de las soluciones analizadas. El modelo de dominio es global, es decir se realiza para todos los casos de uso y no para uno en particular. [18]

El modelo de dominio se realiza si no se logran determinar los procesos de negocio con fronteras bien establecidas donde se logren ver claramente, quienes son las personas que realizan cada proceso de negocio, quienes son los beneficiados con cada uno de estos procesos, pero además quienes son las personas que desarrollan las actividades en cada uno de estos procesos.

El Diseñador de actualizaciones automáticas forma parte de un sistema de actualizaciones que no responde a las exigencias de un cliente, si no que se desarrolla basándose en las experiencias adquiridas por el hecho de las necesidades existentes del centro de salud.

El Diseñador de actualizaciones automáticas tiene como funcionalidad principal la automatización de los procesos de actualizaciones de los distintos proyectos que se desarrollan en el CESIM lo cual no corresponde a ninguno de los procesos de negocio que se contemplan dentro del negocio actual para la realización de dichas actualizaciones.

2.4.1. Conceptos Fundamentales del dominio

PA: La clase PA representa al Proveedor de Actualización que es el único encargado de desarrollar todo el proceso de creación de los paquetes de actualizaciones y realizar las diferentes funcionalidades.

Manifiesto: La clase manifiesto contiene un documento XML con el listado de la información de todos los recursos que el actualizador necesita para actualizar la aplicación y las acciones que este debe realizar.

Paquete de actualización: La clase paquete de actualización es la que contiene los ficheros y un documento XML asociado a una nueva versión. Para cada uno de los paquetes de actualización de entrada existen reglas de validación.

Recursos de Versión: Una clase recursos de versión no es más que aquella que tendrá el control de todas las versiones existentes, paralelo al desarrollo de una actualización.

Directorio: El Directorio de archivos es el ordenador que se fije como repositorio, el mismo contendrá las versiones que se encuentran en una carpeta creada para guardar las mismas, en este directorio se realizarán las salvas continuas de las versiones.

2.4.2. Diagrama de Modelo de Dominio

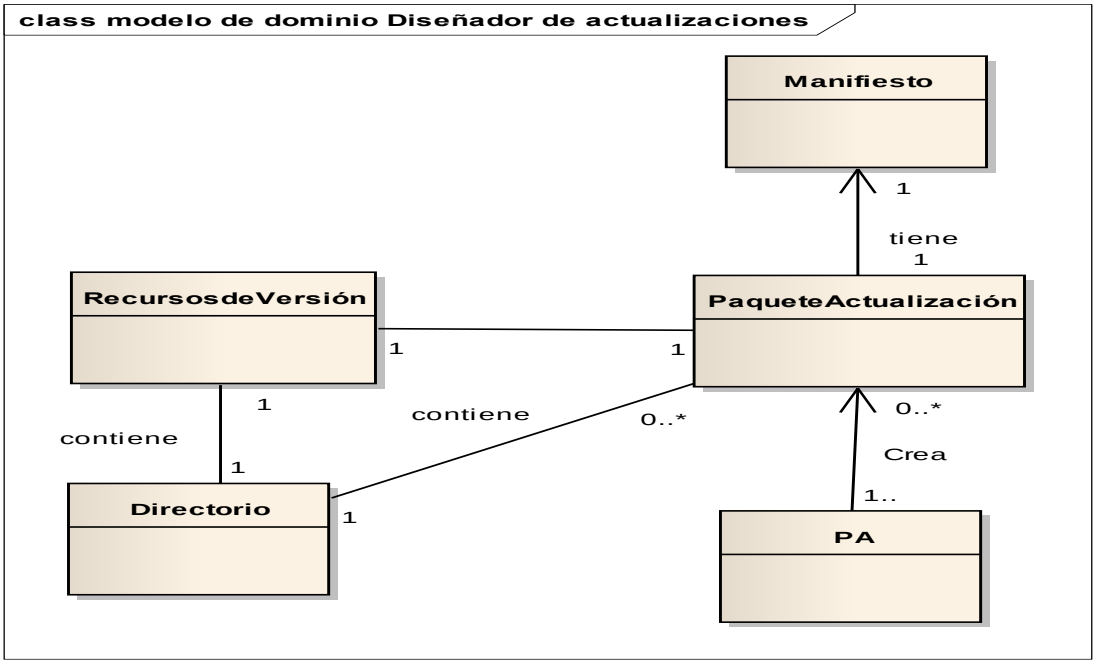

fig. 1 Modelo de dominio.

2.5. Especificación de los requisitos del software.

El flujo de trabajo modelado del negocio da una visión de qué es necesario hacer para dar respuesta a la problemática, lo cual se logra definiendo los procesos más importantes, obteniendo así un modelo de dominio. El modelado del negocio brinda una vía natural para determinar los requerimientos que debe tener el Diseñador de actualizaciones automáticas.

2.5.1. Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Se mantienen invariables sin importar con qué propiedades o cualidades se relacionen.

RF Gestionar producto
- RF1 Insertar producto
- RF2 Eliminar producto

RF Gestionar Perfil
- RF3 Crear perfil de producto
- RF4 Modificar perfil de producto
- RF5 Eliminar perfil de producto

RF Gestionar actualización del producto
- RF6 Crear actualización
- RF7 Modificar actualización
- RF8 Eliminar actualización
- RF9 Insertar descripción de la actualización
- RF10 Publicar actualización
- RF11 Activar publicación
- RF12 Listar publicación
- RF13 Desactivar publicación

2.5.2. Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o

confiable. Son fundamentales en el éxito del producto y normalmente están vinculados a requerimientos funcionales.

Usabilidad

RNF1 El sistema podrá ser utilizado por un Proveedor de actualizaciones, que es el único encargado del diseño y publicación de las actualizaciones que serán utilizadas por el proveedor.

RNF2 El sistema debe garantizar un acceso fácil y rápido.

Seguridad

Confidencialidad

RNF3 La información estará protegida contra accesos no autorizados utilizando mecanismos de autenticación y autorización.

Integridad

RNF5 La información podrá ser modificada solo por personal autorizado.

Disponibilidad

RNF6. El diseñador estará disponible en todo momento, limitando el acceso al sistema solamente por las restricciones que estos tengan de acuerdo a la política de seguridad del sistema.

Eficiencia

RNF7 El diseñador de actualizaciones deberá ser rápido ante la solicitud de desarrollar la nueva versión del producto que se desea actualizar. La seguridad no implicará lentitud o retraso en la repuesta dada por el sistema, por lo que se debe minimizar y reducir el tiempo de respuesta.

Soporte

RNF8 Una vez terminado el diseñador se realizarán las pruebas de integración del mismo con el actualizador, capacitación del personal responsable de la aplicación y mantenimiento de software.

Restricciones de diseño

RNF9 Para el desarrollo se utilizó C# como leguaje de programación.

RNF10 Se utilizó Visual Studio como entorno de desarrollo.

RNF11 Para realizar el modelado del software se utilizó la herramienta Enterprise Architect.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

RNF12 Se dispondrá de la documentación del sistema realizada con la metodología de desarrollo RUP y así generar todo los artefactos correspondientes.

Interfaces de usuario

RNF13 El sistema debe tener una interfaz sencilla, agradable, legible y de fácil uso para el usuario. Debe permitir la ejecución de acciones de manera rápida.

RNF14 El contenido será mostrado de manera comprensible y fácil de leer.

Interfaces Hardware

RNF15 Requerimientos mínimos Ordenador Pentium IV o superior, 512 MB de memoria RAM o superior, Disco duro de 80 GB, copia de seguridad, velocidad del procesador de 2 GHz o superior.

Interfaces Software

RNF16 Garantizar que el diseñador se ejecute sobre Sistema Operativo Windows XP Profesional Service Pack 2.

Estándares Aplicables

RNF17 Para la implementación del diseñador de actualizaciones se deberá seguir los estándares de codificación y las pautas para la documentación, todos definidos por el CESIM.

RNF18 Para las descripciones de casos de uso, y mensajes que debe emitir el sistema, se deben seguir las pautas de análisis definidas en el área temática.

2.6. Modelado del sistema

El modelo del sistema describe cuáles son las personas o grupos de personas, instituciones o sistemas relacionados ya existentes, que interactúan con los procesos que el sistema deberá realizar. Así como la relación que los mismos guardan entre si.

2.6.1. Actores del sistema

A continuación se define y describen los actores del sistema.

Actor	Descripción
PA	El PA es el encargado de realizar todas las operaciones pertinentes a la creación y publicación de las actualizaciones.

2.6.2. Diagrama de Casos de uso del sistema

Un diagrama de casos de uso del sistema representa gráficamente los procesos y su iteración con los actores.

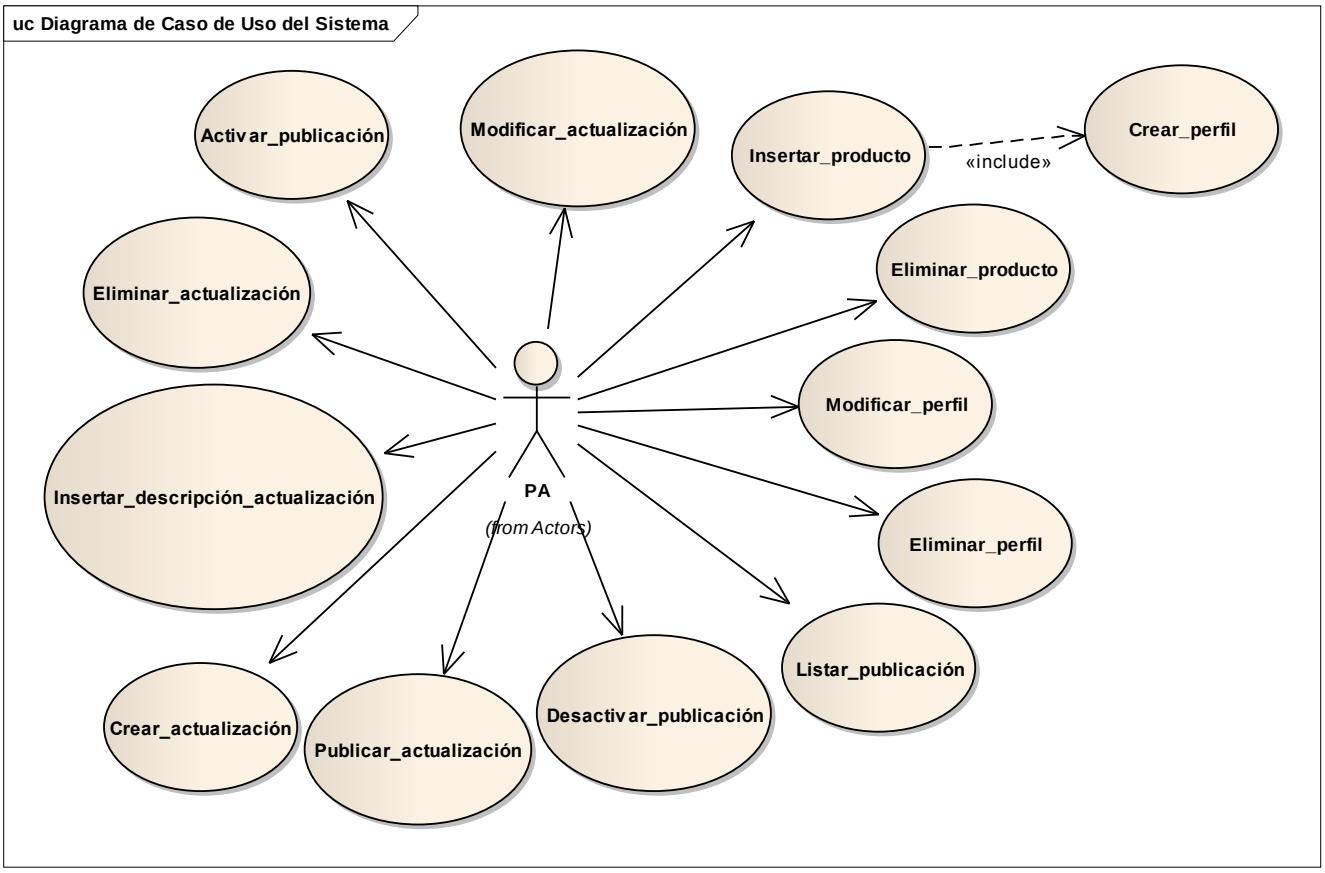

fig. 2 Diagrama de casos de uso del sistema

2.6.3.Especificación de los casos de uso

Descripción del caso de uso: Insertar producto

Objetivo	El objetivo que persigue con este CU es gestionar el producto que se le desea realizar la actualización.
Actores	Proveedor de actualizaciones: (Inicia) Inserta el producto al diseñador.
Resumen	El caso de uso inicia cuando el PA procede a Insertar un producto para realizar la nueva versión del producto, el sistema brinda la posibilidad de eliminar el producto si este no será utilizado, el caso de uso finaliza cuando el PA Inserta el producto.
Complejidad	Media
Prioridad	Crítico

Precondiciones	Al producto necesita una nueva versión.	
Postcondiciones	Se creó un producto.	
Flujo de eventos		
Flujo básico Insertar producto		
	Actor	**Sistema**
1.	El caso de uso inicia cuando el actor accede a inserta el producto al sistema.	
2.		El sistema muestra los datos predeterminados de perfil de producto y las especificaciones del dicho perfil. • Perfil • Licencia • Requerimiento Brinda la posibilidad de introducirlos datos de: • Nombre de la solución • Versión • Licencia Seleccionar: • Plataforma • Velocidad mínima de CPU • RAM • Mínimo de video • Disco Y permite aceptar cada una de las especificaciones del producto.
3.	El actor introduce lo datos de: • Nombre de la solución • Versión • Licencia Selecciona: • Plataforma • Velocidad mínima de CPU • RAM • Mínimo de video • Disco Selecciona la opción de aceptar.	El diseñador permite realizar varias acciones con el producto: - Cancelar operación o eliminar el producto. Ver - Sección 1:”Eliminar producto”.

4.	Crea el release del producto para poder tener todo listo para la actualización.	El sistema muestra todas las acciones relacionadas al release.
5.		El CU termina cuando el sistema muestra el producto seleccionado para realizar actualización.
Sección 1: “Eliminar producto”		
Flujo básico Insertar producto		
	Actor	Sistema
1.	El PA selecciona la opción de eliminar producto.	
2.		El sistema muestra un mensaje “El producto ha sido eliminado satisfactoriamente”.
3.		Regresa a la vista anterior, el CU termina.

Relaciones	CU Incluidos	Insertar producto. Ver CU Crear perfil.
	CU Extendidos	No tiene
Requisitos no funcionales	1,2,5,6	
Asuntos pendientes	Permitir detener en el momento que no se desee continuar con la inserción del producto.	

Descripción del caso de uso: Crear perfil

Objetivo	El objetivo que persigue con este CU es crear el perfil del producto para que se pueda llevar a cabo la nueva actualización con los requisitos específicos.
Actores	Proveedor de actualizaciones: (Inicia) Crea el perfil del producto.
Resumen	El caso de uso inicia cuando el PA accede a la opción de Crear perfil del producto, el sistema brinda la opción de introducir los datos del perfil, el sistema crea el perfil, el caso de uso finaliza.
Complejidad	Media
Prioridad	Crítico
Precondiciones	Al producto ha sido insertado al diseñador.
Postcondiciones	Se creó el perfil del producto.
Flujo de eventos	
Flujo básico Crear perfil	

	Actor	**Sistema**
1.	El caso de uso inicia cuando el PA accede a la	

	opción de crear el perfil.	
2.		El sistema muestra los datos predeterminados de perfil de producto y las especificaciones del dicho perfil. • Perfil • Licencia • Requerimiento Brinda la posibilidad de introducirlos datos de: • Nombre de la solución • Versión • Licencia Seleccionar: • Plataforma • Velocidad mínima de CPU • RAM • Mínimo de video • Disco Y permite aceptar cada una de las especificaciones del producto.
3.	El actor introduce lo datos de: • Nombre de la solución • Versión • Licencia Selecciona: • Plataforma • Velocidad mínima de CPU • RAM • Mínimo de video • Disco Selecciona la opción de aceptar.	
4.		El sistema muestra que su perfil es completo.
5.		El diseñador permite realizar varias acciones con el perfil: - Eliminar perfil. Ver - Sección 1:”Eliminar perfil”. -
6.		El CU termina cuando el sistema muestra el perfil de producto con todos los datos listo

		para realizar la nueva versión.
Sección 1: “Eliminar perfil”		
Flujo básico Crear perfil		
	Actor	Sistema
1.	El PA selecciona en el menú Eliminar perfil del producto.	
2.		El sistema muestra un mensaje ¨El perfil del producto ha sido eliminado satisfactoriamente¨.

Relaciones	CU Incluidos	Este es un caso de uso incluido ya que no se podrá ejecutar al menos que se halla insertado el producto y siempre que este sea insertado se creará su respectivo perfil. Crear perfil. Ver CU Insertar producto.
	CU Extendidos	No tiene
Requisitos no funcionales	1,2,5,6	
Asuntos pendientes	Permitir que al crear el perfil se gestiones más especificidades del producto.	

Descripción del caso de uso: Crear actualización

Objetivo	El objetivo de crear la actualización consiste en construir un paquete con la nueva versión y generar el manifiesto de actualización.
Actores	Proveedor de actualizaciones: (Inicia) Crea la actualización.
Resumen	El caso de uso inicia cuando el PA accede a la opción de crear la nueva versión del producto, el sistema brinda la posibilidad de introducir los datos para la actualización, el PA introduce los datos de la actualización, el caso de uso finaliza cuando el PA crea la actualización del producto.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Existe el producto y se ha creado su perfil.
Postcondiciones	Se ha creado la actualización del producto.
Flujo de eventos	
Flujo básico Crear actualización	

	Actor	**Sistema**
1.	El caso de uso inicia cuando el PA accede a la opción de crear la nueva versión de un producto.	
2.		Se muestran los datos predeterminados

		de: • Adicionar eventos de actualización • Añadir elementos de registro • Adicionar acciones a ficheros Brinda la posibilidad de introducir los datos de la nueva versión. • Nombre del proceso • Servicios • Clave • Valor • Dato • Nombre Seleccionar: • Archivo • Tipo • Evento • Acción Y permite: Aceptar crear la versión.
3.	Introduce los datos de crear la actualización. • Nombre del proceso • Servicios • Clave • Valor • Dato • Nombre Selecciona: • Archivo • Tipo • Evento • Acción Selecciona la opción de aceptar cada uno de los eventos para crear la actualización.	
4.		Brinda la posibilidad de introducir: • Descripción de la actualización Permite: • Aceptar crear descripción de la actualización

5.	El actor inserta la descripción de la actualización.	
6.		El sistema genera el manifiesto.
7.		El sistema crea un compactado en formato .ZIP.
8.		Calcula el valor hash y el tamaño del compactado.
9.		El diseñador permite realizar varias acciones con la actualización: - Eliminar actualización. Ver - Sección 1:"Eliminar actualización".
10		El CU termina cuando el sistema manda un mensaje "El documento de tareas de ficheros creado con éxito".

Sección 1: "Eliminar actualización"		
Flujo básico Crear actualización		
	Actor	Sistema
11.	El PA selecciona en el menú la opción Eliminar actualización.	El sistema muestra un mensaje "La actualización del producto ha sido eliminada satisfactoriamente".

Relaciones	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales	1,2,3,4,5	
Asuntos pendientes	La actualización a de ser publicada.	

Descripción del caso de uso: Publicar actualización

Objetivo	El objetivo de publicar la actualización consiste en que la nueva versión del producto para ser utilizada por el actualizador automático a de ser publicada.
Actores	Proveedor de actualizaciones: (Inicia) Publica la actualización.
Resumen	El caso de uso inicia cuando el PA accede a la opción publicar la actualización de un producto, el sistema brinda la posibilidad de introducir los datos para publicar en el servidor, el PA introduce los datos de la publicación, el caso de uso finaliza cuando el PA publica la nueva actualización del producto.
Complejidad	Media
Prioridad	Crítico

Precondiciones	Existe la actualización y se ha actualizado el índex.	
Postcondiciones	Se publicó la actualización.	
Flujo de eventos		
Flujo básico Publicar actualización		
	Actor	**Sistema**
1.	El caso de uso inicia cuando el PA accede a la opción de publicar.	
2.		El sistema muestra los datos de: • Servidor • Índex Brinda la posibilidad de introducir los datos de la publicación. • Nombre • Dirección • Puerto • Usuario • Contraseña Seleccionar: • Protocolo Y permite: Aceptar publicar.
3.	Introduce los datos de la publicación. • Nombre • Dirección • Puerto • Usuario • Contraseña Selecciona: • Protocolo Selecciona la opción de aceptar para publicar la actualización.	
4.	Analiza los archivos y conecta con el servidor.	Servidor conectado.
5.	Analiza los archivos de las plataformas y descarga el índex.	El sistema actualiza el índex.
6.	Sube el producto con la versión del producto.	
7.		El diseñador permite realizar varias acciones

		con la actualización: - Activar publicación. Ver - Sección 1:" Activar publicación". - Desactivar publicación. Ver - Sección 2:" Desactivar publicación".
8.		El CU termina cuando el sistema lanza un mensaje ¨se ha publicado actualización¨.
Sección 1: "Activar publicación"		
Flujo básico Publicar actualización		
	Actor	Sistema
1.	El PA selecciona la opción de activar publicación.	
2.		El sistema activa la nueva versión existente si esta se encuentra desactivada.
Sección 1: "Desactivar publicación"		
Flujo básico Publicar actualización		
	Actor	Sistema
1.	El PA selecciona la opción desactivar.	
2.		El sistema desactiva automáticamente dicha versión del producto.

Relaciones	CU Incluidos	No tiene
	CU Extendidos	No tiene
Requisitos no funcionales	1,2,3,4,5,6	
Asuntos pendientes	La actualización ha de ser publicada.	

Definiciones de diseño que se aplican:

El diseño de la interfaz es otro de los puntos fundamentales a tratar a la hora de la presentación de la aplicación teniendo en cuenta que esa es la capa de presentación al usuario y por lo tanto debe ser lo más amigable y comprensible, por lo que debe resultar coherente, sensata y necesaria para el usuario.

En este caso la aplicación está dirigida mayormente a un informático que será el encargado de realizar todos los procesos referente a las actualizaciones, teniendo esto en cuenta, los esfuerzos de diseño están orientados a lograr una interfaz clara y fácil de usar, ya que a través de la misma es que se interactúa con los recursos que se ponen a disposición del usuario en el ordenador. Por tanto, el desarrollo de la aplicación se basa en algunos de los Principios del Diseño Universal o Diseño para Todos, estos principios se centran en el diseño de aplicaciones teniendo en cuenta la cultura, el conocimiento y el ambiente, que influyan sobre los usuarios a los que va dirigido el producto y se presentan a continuación:

- ✓ **Flexibilidad**: El diseño debe poder adecuarse a un amplio rango de preferencias y habilidades individuales.
- ✓ **Información fácil de percibir**: El diseño debe ser capaz de intercambiar información con usuario, independientemente de las condiciones ambientales o las capacidades sensoriales del mismo.
- ✓ **Tolerante a errores**: El diseño debe minimizar las acciones accidentales o fortuitas que puedan tener consecuencias fatales o no deseadas.
- ✓ **Escaso esfuerzo físico**: El diseño debe poder ser usado eficazmente y con el mínimo esfuerzo posible.
- ✓ **Dimensiones apropiadas**: Los tamaños y espacios deben ser apropiados para el alcance, manipulación y uso por parte del usuario, independientemente de su tamaño, posición, y movilidad.

En este capítulo se identificaron los principales conceptos del Diseñador de Actualizaciones Automáticas, mostrándose además las descripciones de cada una de las clases que intervienen en el modelo de dominio. Se presenta también un listado de los requerimientos funcionales y no funcionales que indican las capacidades o condiciones que el sistema debe cumplir y las propiedades o cualidades que el producto debe tener. Por último se conforman los casos de uso del sistema, y estos son descritos al igual que sus actores.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

Este capítulo profundiza el análisis y diseño del sistema, donde mediante diagramas se llevará a cabo todo el flujo de los procesos. A través de los diagramas de interacción del sistema, se describe gráficamente la interacción entre los actores y el sistema, quedando reflejados los mensajes que se transmiten entre los objetos. Además se describen las clases de diseño. Se plantean la seguridad en el sistema, y la forma en que está concebida que se despliegue la aplicación.

1.1 Análisis

En esta etapa se refinan y estructuran los requisitos obtenidos con anterioridad, profundizando en el dominio de la aplicación lo que permitirá una mayor comprensión del problema para modelar la solución. Los diagramas representados están organizados por paquetes para una mejor comprensión de los mismos.

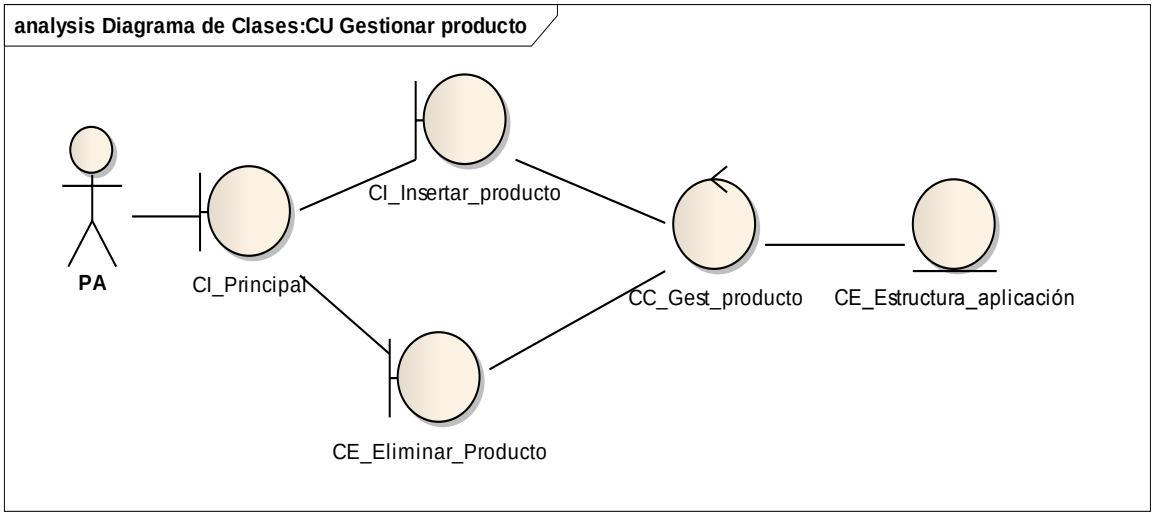

Fig.3 Diagrama de clases del análisis del CUS Gestionar producto

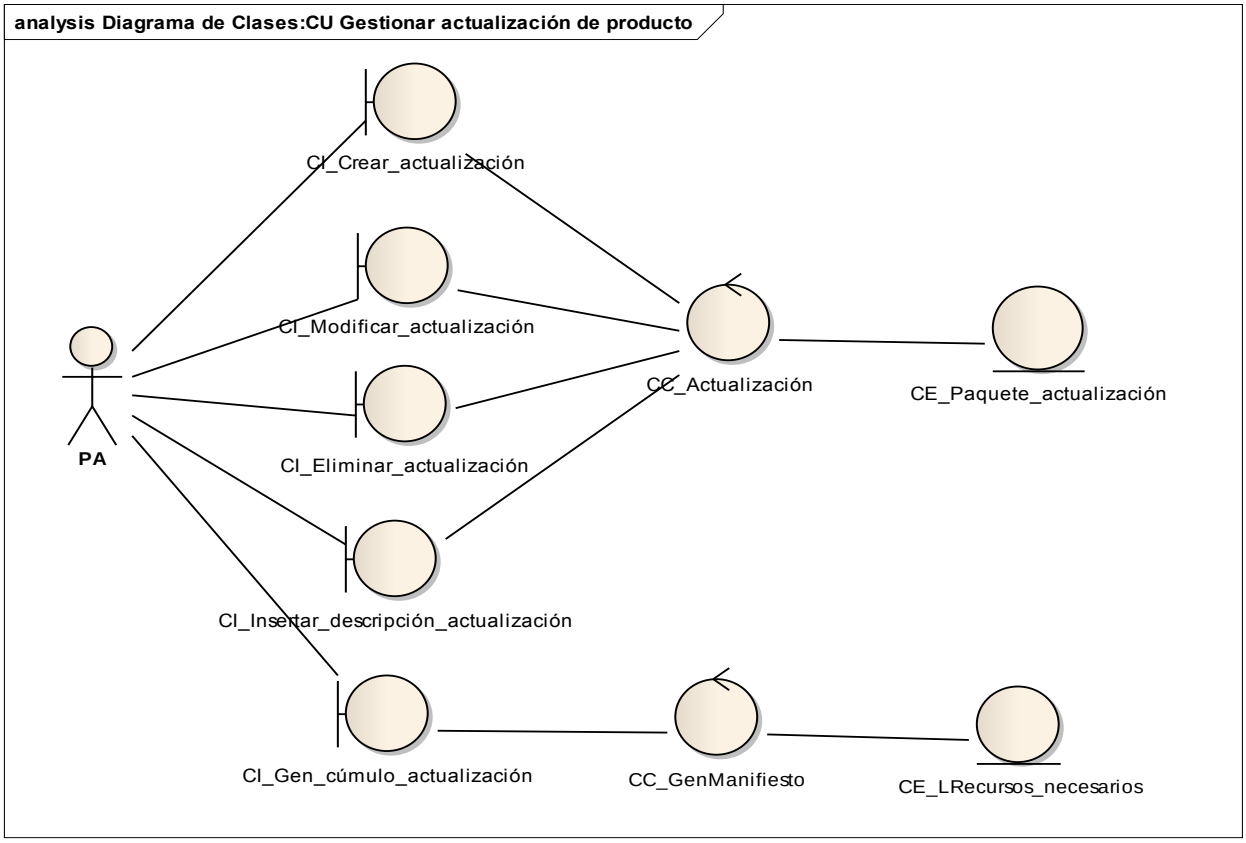

Fig. 4 Diagrama de clases del análisis del CUS Gestionar actualización de producto

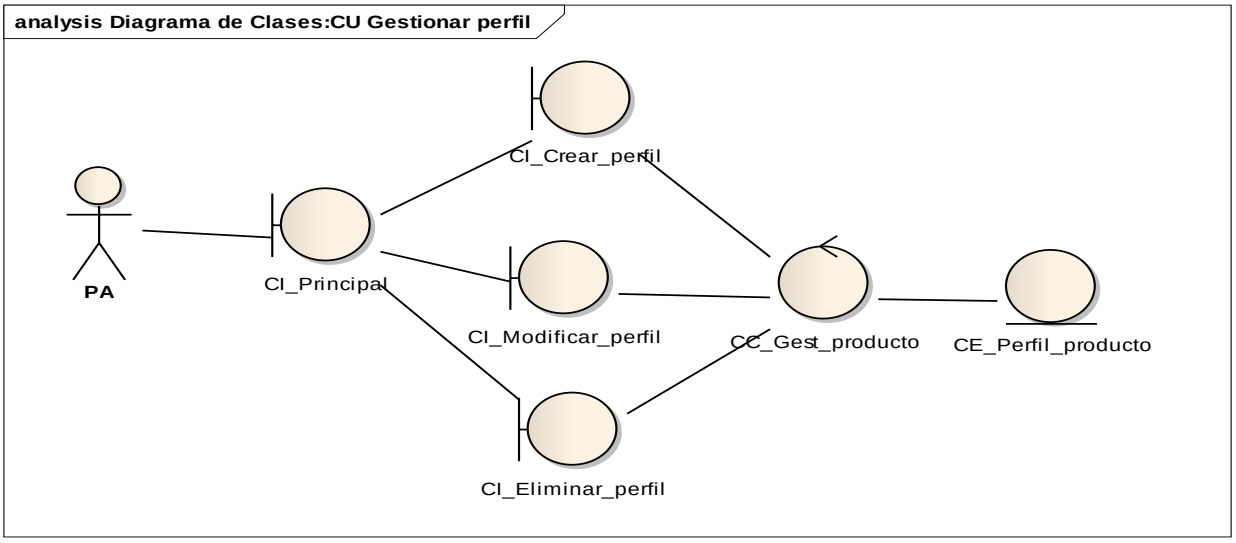

Fig. 5 Diagrama de clases del análisis del CUS Gestionar perfil

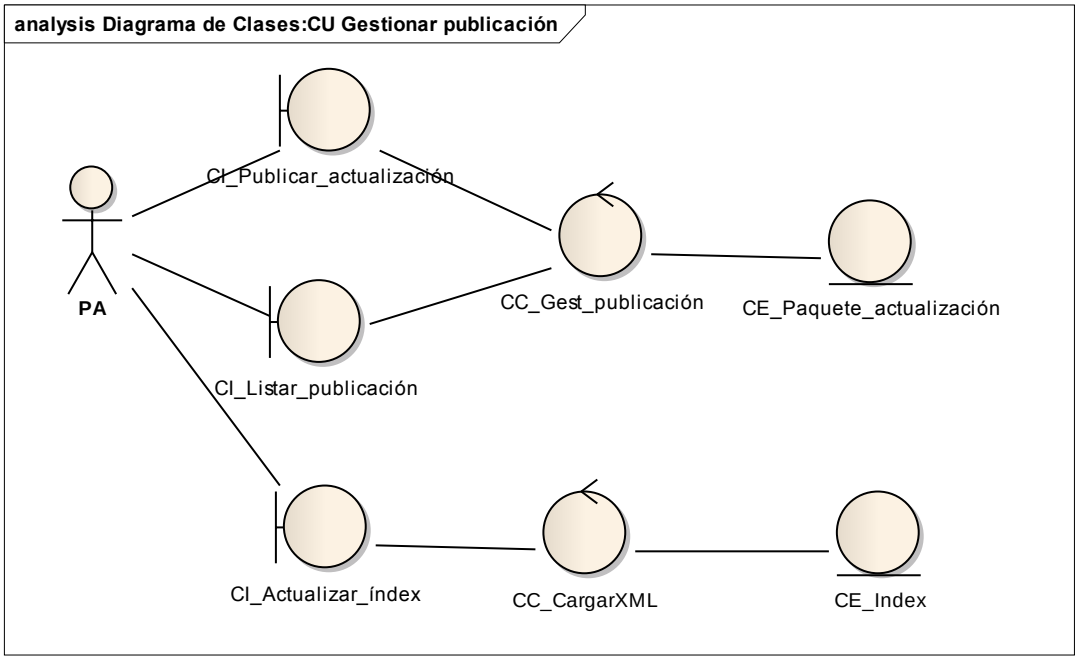

Fig. 6 Diagrama de clases del análisis del CUS Gestionar publicación

1.2 Estructura del diseño

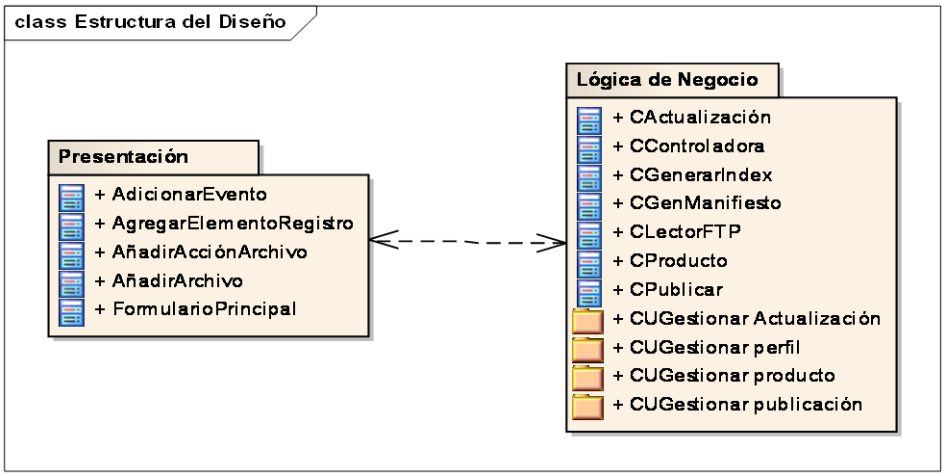

Fig. 7 Estructura del diseño

Capa de Presentación

En la capa de presentación se encuentran los formularios cuyas funcionalidades consisten en intercambiar información con el proveedor de actualización, o sea, son las únicas clases con la que interactúa directamente el usuario. Estas clases contienen funciones mediante las cuales se invocan los métodos de las clases que se encuentran en la capa lógica de negocio.

Capa Lógica de Negocio

En la capa de lógica de negocio se encuentran las clases que establecen la comunicación entre la capa de presentación y la capa de datos, encargadas de recibir y responder cada petición del PA. Estas clases reciben las solicitudes de los clientes, actualizando o recuperando información y emitiendo una respuesta.

1.3 Diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en los requisitos no funcionales y otras restricciones del entorno de implementación que tienen impacto en el sistema, por tanto define las clases del diseño que conformarán el sistema a implementar.

1.3.1 Diagrama de clases del diseño por Caso de Uso

Los diagramas de clases del diseño describen gráficamente las especificaciones de las clases del software y contienen las clases, atributos, métodos, navegabilidad y dependencias existentes entre ellas. A continuación se representan los diagramas de clase del diseño por casos de uso del sistema.

CUS Gestionar producto

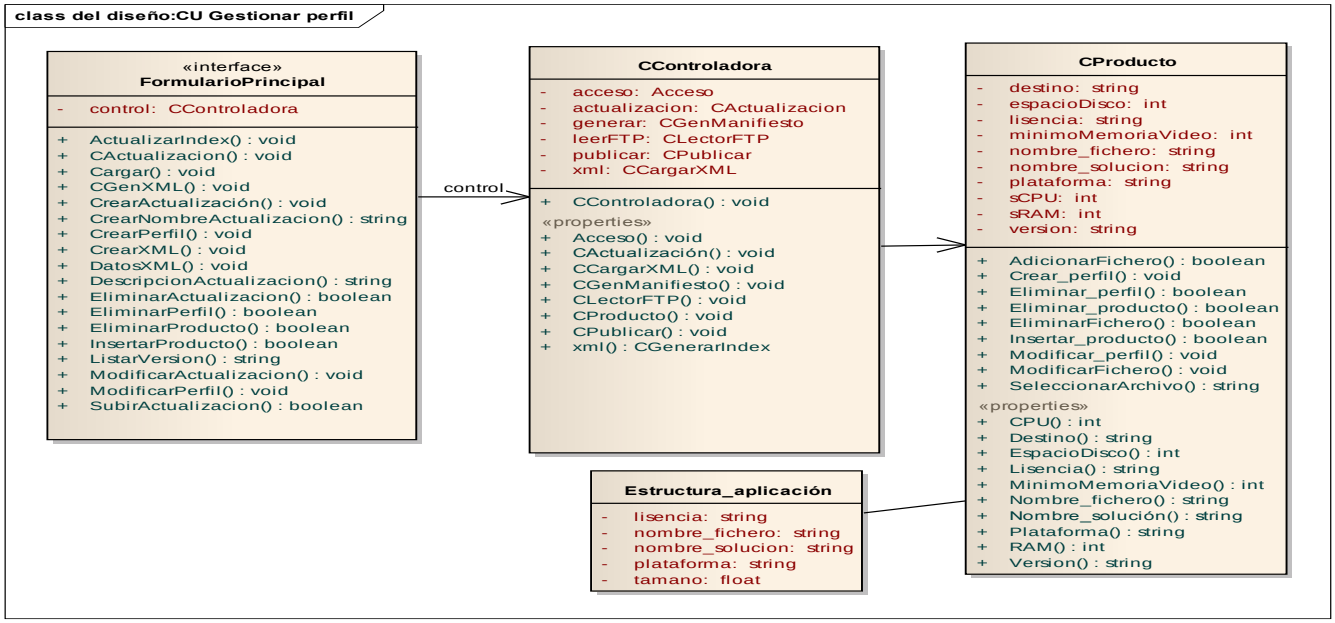

Fig. 8 Diagrama de clases del diseño del CUS Gestionar producto

CUS Gestionar perfil

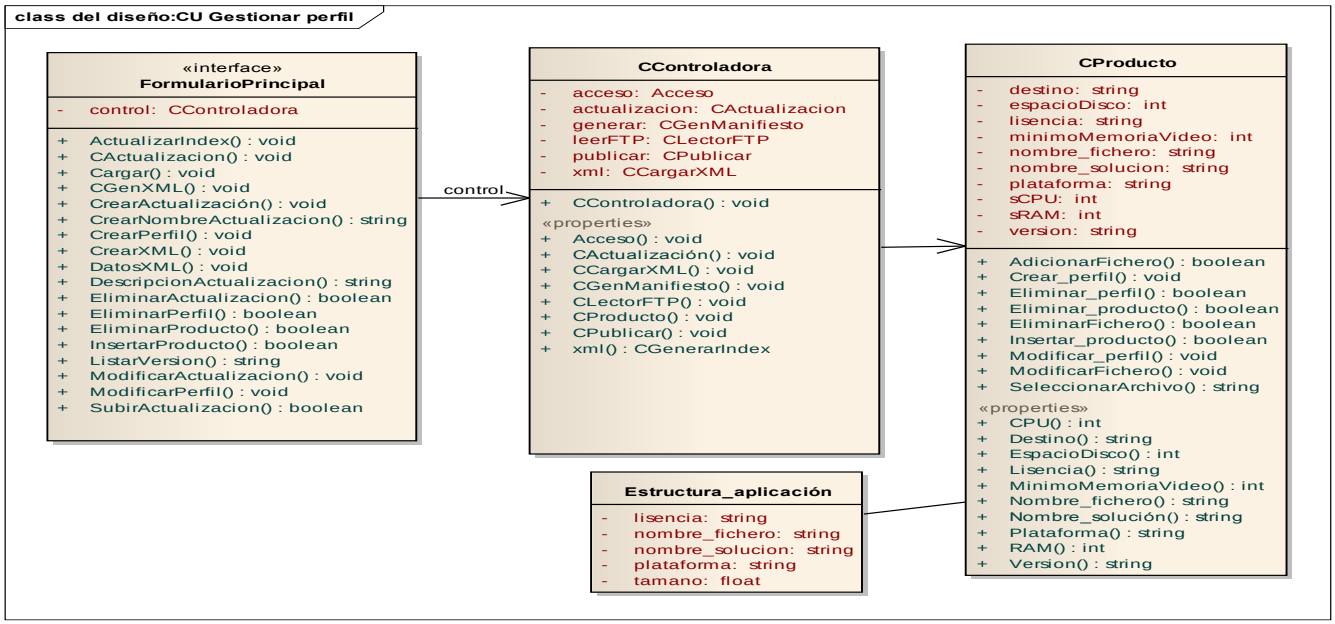

Fig. 9 Diagrama de clases del diseño del CUS Gestionar perfil

CUS Gestionar publicación

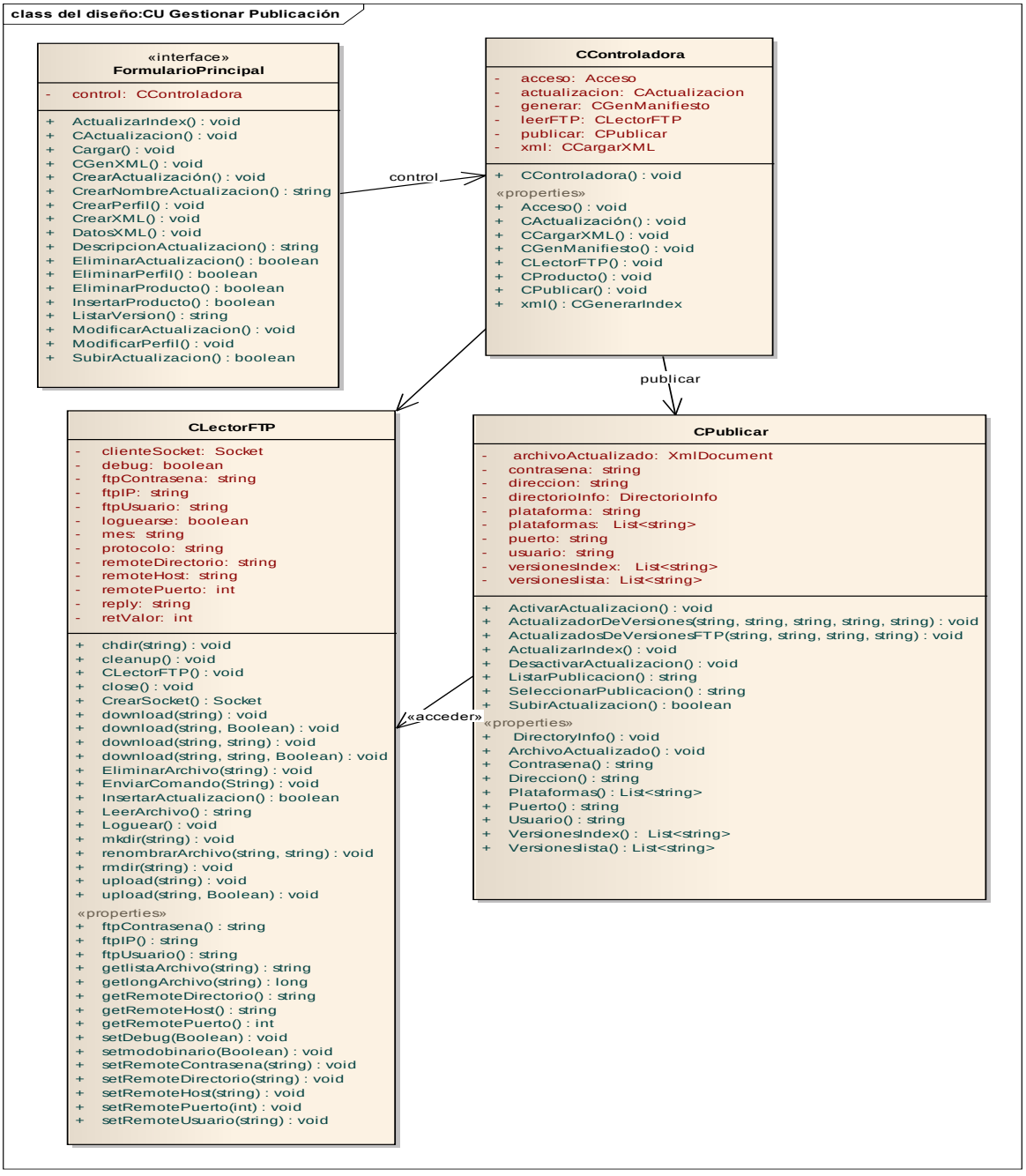

Fig. 10 Diagrama de clases del diseño del CUS Gestionar publicación

CUS Gestionar Actualización

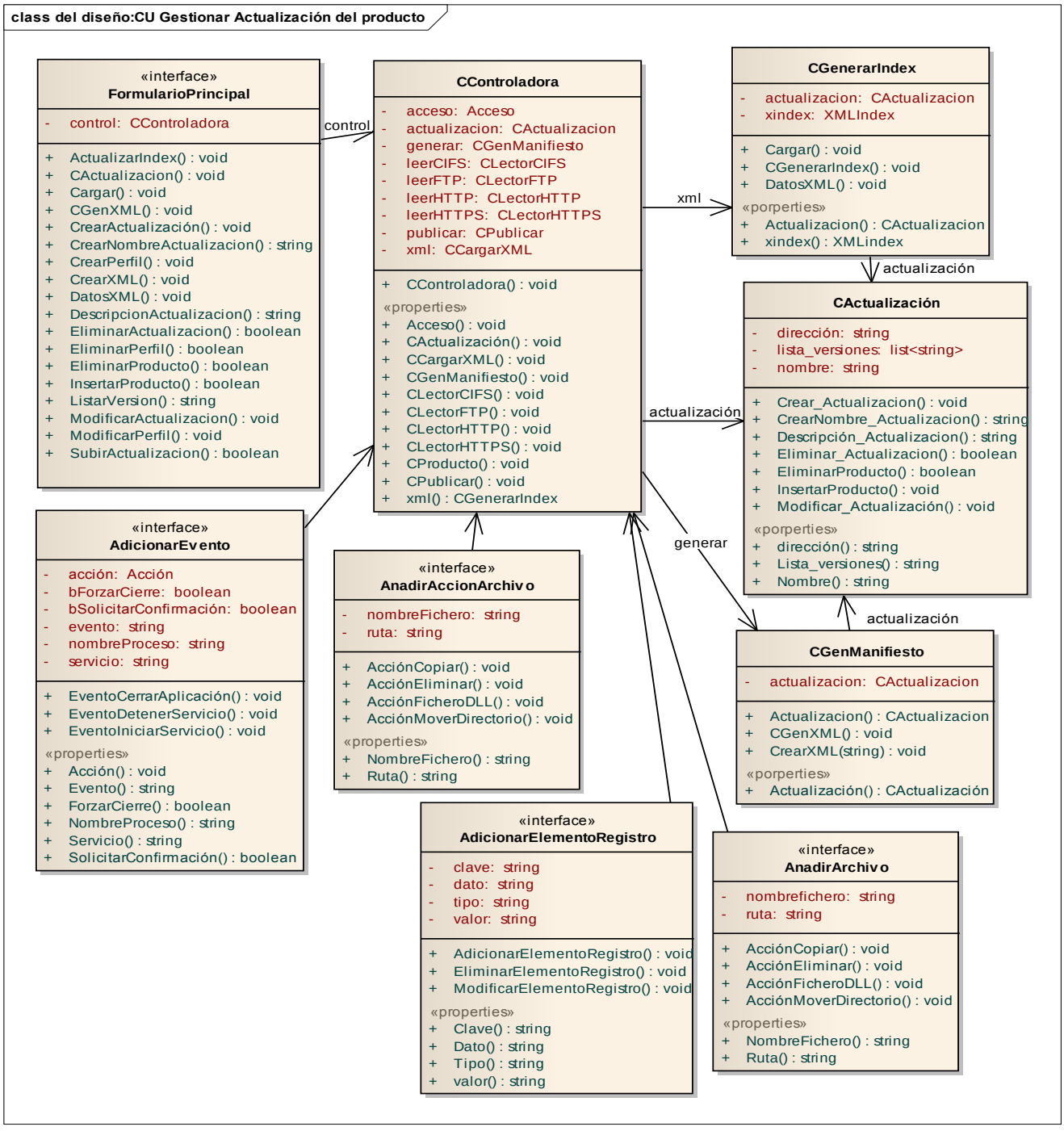

Fig. 11 Diagrama de clases del diseño del CUS Gestionar actualización de producto

1.3.2 Diagramas de secuencia

Los diagramas de secuencia muestran los objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos. En los mismos se puede observar las interacciones entre objetos, ordenadas en una secuencia de tiempo. Seguidamente se muestran los diagramas de secuencia por casos de uso del sistema. Ver anexo 2.

1.4 Descripción de las clases

Capa de presentación

Las funcionalidades de la capa de presentación consisten en intercambiar información con los actores, por lo tanto es la única capa con la que interactúa directamente el usuario.

Nombre: FormularioPrincipal
Descripción: Esta es la interfaz principal del Diseñador sobre la cual el usuario realizará todas las operaciones pertinentes a una actualización, pertenece a la capa de presentación. En esta interfaz se encuentran los siguientes menús: Gestionar Productos, Gestionar Actualización, Administrar Usuarios, Publicar Actualización; en cada menú existen varias opciones que se corresponden con las funcionalidades del Diseñador de actualizaciones.

Nombre: AdicionarEvento
Descripción: Es el formulario donde el proveedor de actualizaciones introduce los pasos necesarios que se deben realizar antes o después de una actualización de software.

Nombre: AgregarElementoRegistro
Descripción: Es el formulario donde el proveedor de actualizaciones introduce las nuevas llaves que se van a insertar al registro del sistema.

Nombre: AñadirArchivo
Descripción: Es el formulario donde se asigna un archivo a una acción determinada.

Nombre: AñadirAcciónArchivo
Descripción: Es el formulario donde el proveedor asigna una acción determinada a un archivo, es donde se describe si se registran o no los archivos .dll después de habérsele asignado alguna acción y además se permite mover un directorio de la estructura del árbol de directorios.

Capa de Negocio

Se comunicación con la capa de presentación y es la encargada de recibir y responder cada petición de los usuarios.

Nombre: CControladora
Descripción: Esta clase es la encargada de manejar todas las clases del negocio por lo que maneja toda la información del Diseñador.

Nombre: CGenManifiesto
Descripción: Las funcionalidades de esta clase consisten en generar de un paquete de actualización creado el manifiesto con sus tareas correspondientes.

Nombre: CCargarXML
Descripción: Las funcionalidades de esta clase consisten en cargar un XML que se encuentra en el servidor con los datos de los paquetes y actualizarlos.

Nombre: CActualización
Descripción: Esta clase contiene todo la información correspondiente a las actualizaciones de los productos.

Nombre: CPublicar
Descripción: En esta clase se encarga de actualizar la información en el servidor y controlar la misma según el tipo de protocolo.

Nombre: CProducto
Descripción: En esta clase se gestiona todo lo referente al producto que se le desea realizar la actualización.

Nombre: CLectorFTP
Descripción: Esta clase se encarga de la conexión al servidor y sus funciones mediante el protocolo FTP.

En este capítulo, se obtuvo el diagrama de clases del diseño para cada caso de uso del sistema, donde se representaron las relaciones que se establecen entre las clases. Además quedaron definidos los diagramas de interacción y al modelar el sistema se tuvieron en cuenta todos los requisitos previstos y las restricciones que estos deben cumplir.

CAPÍTULO 4: IMPLEMENTACIÓN

En el presente capítulo tiene como objetivo describir el modelo de implementación teniendo en cuenta el resultado obtenido en el diseño. Se modela el sistema en términos de componentes y este se organiza de acuerdo a los nodos específicos que conforman el modelo de despliegue. Se incluye en el desarrollo del capítulo los diagramas de componentes y despliegue.

2.1 Modelo de implementación

El modelo de implementación describe como los elementos del modelo de diseño y las clases, se implementan en términos de componentes, ficheros de código fuente, ejecutables, entre otros. Los diagramas de despliegue y componentes conforman lo que se conoce como un modelo de implementación, al describir los componentes y construir su organización y dependencia entre los nodos físicos en que funcionará la aplicación.

2.2 Componentes

Un componente es una parte física y reemplazable de un sistema que se conforma con un conjunto de interfaces y proporciona la realización de dicho conjunto. Representa una unidad de código (fuente, binario o ejecutable) que permite mostrar las dependencias en tiempo de compilación y ejecución. Las instancias de componentes de software muestran unidades de software en tiempo de ejecución y generalmente ayudan a identificar sus dependencias y su localización en nodos.

2.3 Diagrama de componentes

Un Diagrama de Componentes ilustra los fragmentos de software que conformarán un sistema. Un diagrama de componentes tiene un nivel de abstracción más elevado que un diagrama de clase, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, como así eventualmente un componente puede comprender una gran porción de un sistema.

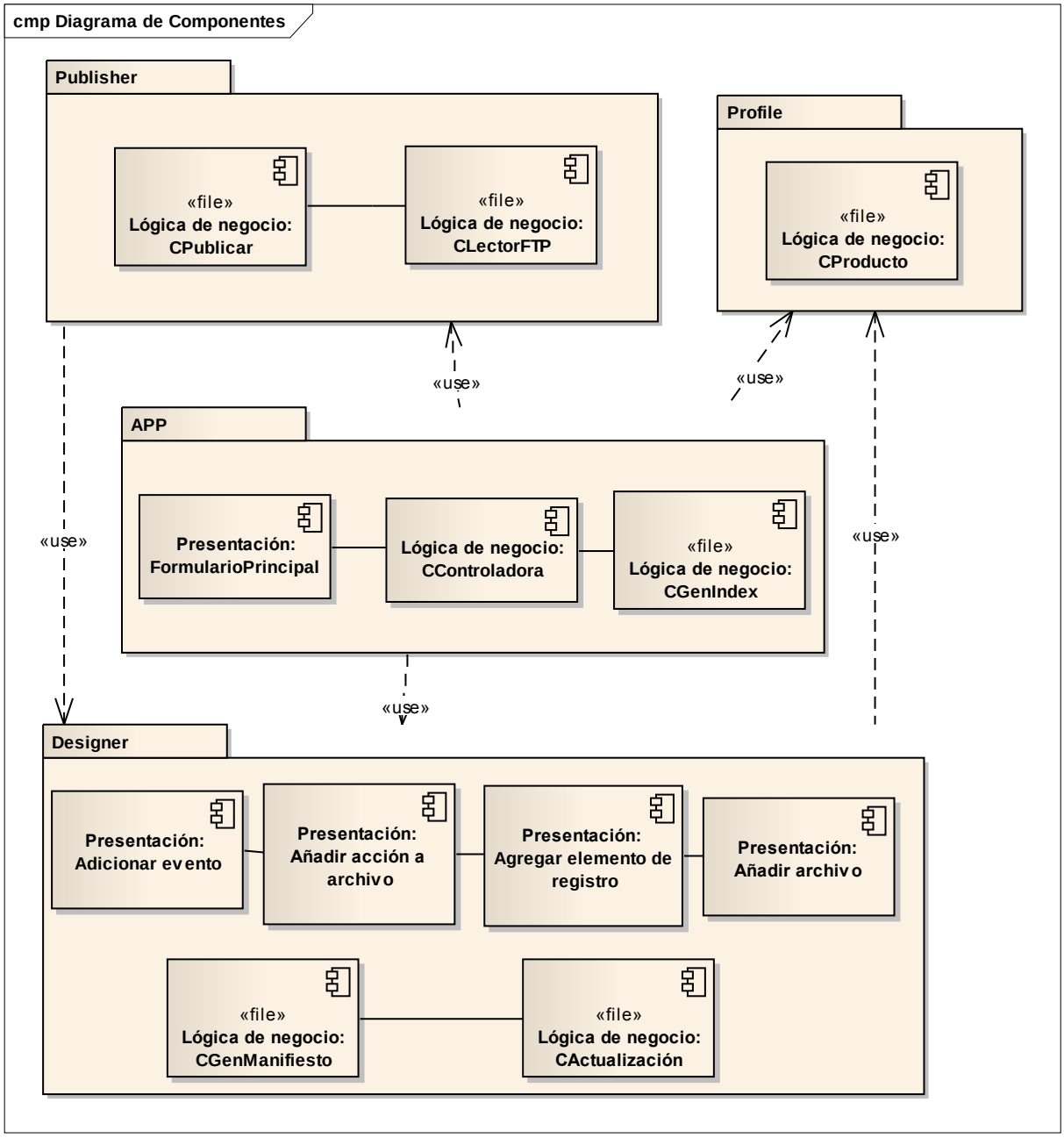

Fig. 22 Diagrama de Componentes

2.4 Diagrama de despliegue

Un Diagrama de Despliegue muestra cómo y dónde se desplegará el sistema. Las máquinas físicas y los procesadores se representan como nodos. Como los artefactos se ubican en los nodos para modelar el despliegue del sistema, la ubicación es guiada por el uso de las especificaciones de despliegue.

Para desplegar el Diseñador de actualizaciones automáticas se requiere una PC cliente donde se instalará el Diseñador de actualizaciones, una PC servidora donde se encuentra publicadas las actualizaciones de los productos.

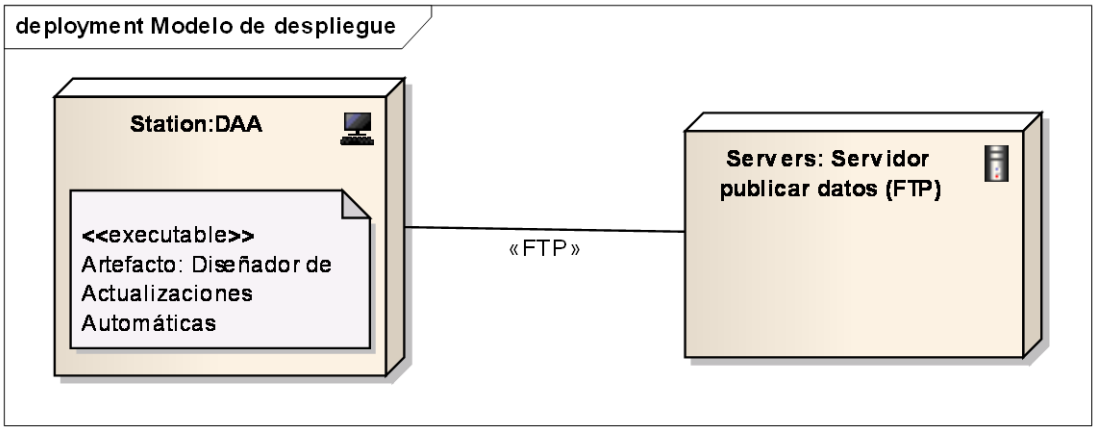

Fig. 23 Diagrama Despliegue

2.5 Restricciones de nomenclatura y Estándares de Codificación

Idioma: Se debe utilizar como idioma el español, las palabras no se acentuarán.

Identación **Objetivo:** Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento.		
Inicio y fin de bloque		Se recomienda dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach.
Aspectos Generales	El identado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la PC o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deber estar alineados debajo de la	

	declaración a la que pertenecen y deben evitarse si hay sólo una instrucción. Nunca colocar {en la línea de un código cualquiera, esto requiere una línea propia.
Comentarios, separadores, líneas, espacios en blanco y márgenes. ***Objetivo:*** Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez.		
Ubicación de comentarios	Al inicio de cada clase o función y al final de cada bloque de código.	Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas.
Líneas en blanco	Se emplean antes y después de métodos, clases y estructuras.	Se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función
Espacios en blanco	Entre operadores lógicos y aritméticos.	Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código. Ejemplo: producto = nomproducto
Aspectos generales	Sobre el comentario	Se debe evitar comentar cada línea de código. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción.
	Sobre los espacios en blanco	No se debe usar espacio en blanco: Después del corchete abierto y antes del cerrado de un arreglo. Después del paréntesis abierto y antes del cerrado. Antes de un punto y coma.
Variables y constantes		
Apariencia de variables	Las variables tendrán un prefijo para el tipo de datos en minúscula.	El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, la cual identificará el tipo de datos al que se refiere (tabla 1.1), en caso de que sea un nombre compuesto se empleará notación CamellCasing[1]. Ejemplo: sNombrePaciente
Apariencia de constantes	Todas sus letras en mayúsculas.	Se deben declarar las constantes con todas sus letras en mayúscula.
Aspectos generales	Nombres de las variables y constantes.	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.
Clases y Objeto		

[1] **Notación CamellCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula excepto la primera palabra que debe iniciar con minúscula.
Ejemplo: notacionCamelCasing.

Objetivo: Nombrar las clases e instancias de forma estándar para todas las aplicaciones.		
Apariencia de clases y objetos	Primera letra en mayúscula.	Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing[2]. Ejemplo: MiClase(). Para el caso de las instancias se comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.
Apariencia de atributos	Primera letra en minúscula	El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto se empleará notación CamellCasing.
Apariencia de las funciones	Primera letra en mayúscula	Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hace la función. Se empleará notación PascalCasing. Ejemplo: function BuscarUnidad(). Si son funciones que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set
Declaración de parámetro en funciones	Agrupados por tipos Poner los string 1 numéricos 2, además, agrupar según valores por defecto.	Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que estén agrupados por el tipo de dato que contienen, especificando el tipo de datos (tabla 1.1).
Aspectos generales	Sobre las clases, los objetos, los atributos y las funciones.	El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con solo leerlo se conozca el propósito de los mismos.
		Controles
Apariencia de los controles.	Los controles tendrán un prefijo para el tipo de datos en minúscula.	El nombre que se le da a los controles deben comenzar con las primeras letras en minúscula, las cuales identificaran el tipo de datos al que se refiere (tabla 1.2), en caso de que sea un nombre compuesto se empleará notación CamellCasing. Ejemplo: btnAcepta. [19]

Tabla 1.1

Tipo Datos	Prefijo	Ejemplo
Int	i	iCantPacientes
flota	f	fPesoPaciente

[2] **Notación PascalCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula.
Ejemplo: NotacionPascalCasing

double	d	dPesoCarro
bool	b	bPacienteActivo
string	s	sNombrePaciente
char	c	cLetra
De tipo enum	ev	evSexo
byte	b	bCantDiasPaciente
sbyte	sb	sbEdadPaciente
short	sh	shVariableShort
ushort	us	usVariableUshort
uint	ui	uiVariableUint
long	l	lVariableLong
ulong	ul	ulVariableUlong
decimal	dc	dcVariableDecimal
Objetos	o	oPacienteHistorico
Objetos de tipo Struct	st	stUnaStruct

Tabla 1.2

Control	Prefijo	Ejemplo
Botón	btn	btnAceptar
Etiqueta	lbl	lblNombre
Lista/Menú	mn	mnPrincipal
Campo de Texto	txt	txtFecha
Casilla de Verificación	chx	chxBorrar
Casilla de Selección	cbx	cbxSexo

En este capítulo, se obtuvieron los diagramas de componentes y de despliegue, mostrándose cómo se encuentra divido por paquetes de clase el desarrollo del Diseñador agrupando clases a fines según la capa a la que pertenecen, presentación y negocio. Se especifica las relaciones que existen entre los componentes que indican el grado de interrelación que existe entre ellos. Además se definen los estándares de codificación que se siguieron para la implementación.

CONCLUSIONES

Una vez concluida la investigación sobre las actualizaciones automáticas y su necesidad en el Centro de Informática Médica y el desarrollo del Diseñador de Actualizaciones Automáticas, se ha dado cumplimiento a los objetivos planteados obteniéndose las siguientes conclusiones:

- El estudio realizado sobre las principales tendencias y tecnologías informáticas actuales para el desarrollo de la aplicación, evidenció que estos no cubren todas las necesidades para el diseño y publicación de las actualizaciones de los productos elaborados en el Centro de Informática Médica.
- La aplicación desarrollada contiene las principales funcionalidades asociadas al dominio de los diseñadores de actualizaciones, logrando así un sistema acorde a las tendencias actuales en este domino.
- Se disminuye considerablemente el tiempo consumido por el proceso de diseño de las actualizaciones automáticas.
- La aplicación creada garantiza la gestión de las actualizaciones para productos en cualquier ambiente o plataforma de software, además elimina la introducción de errores en el proceso de diseño y empaquetamiento de las actualizaciones.
- La solución desarrollada servirá de referencia para el desarrollo sistemas similares y constituye la base para la extensión de los productos desarrollados en el Centro de Informática Médica.

RECOMENDACIONES

Para siguientes fases de desarrollo de esta solución de actualización automática de aplicaciones, se han definido un conjunto de funcionalidades y herramientas que se recomienda implementar. De esta forma aumentará su utilidad en los distintos entornos y condiciones donde se pueda desplegar, así como lograr mayores facilidades al usuario administrador para el control y configuración de la actualización. Por la experiencia adquirida durante el desarrollo de la presente investigación, los autores recomiendan:

- Implementar un método de acceso que garantice que los proveedores de actualizaciones sólo puedan modificar la información de los productos que estos suministran.
- Implementar una solución que garantice el acceso concurrente al servidor para la publicación de actualizaciones.
- Extender el perfil del producto para registrar la estructura del árbol de directorios de la aplicación.
- Establecer una política para la creación de productos en el CESIM, de forma que se facilite el proceso de creación de las actualizaciones para estos.

REFERENCIAS BIBLIOGRÁFICAS

[1] Conceptos básicos de Computación. [Online] [Citado: Enero 20,2010] Disponible en http://diccionario.babylon.com/aplicación informática

[2] Conceptos básicos de Computación. [Online] [Citado: Enero 20,2010] Disponible en http://www.xuletas.es/ficha/conceptos-basicos-de-computacion/

[3] Diccionario de la lengua española. [Online] [Citado: Enero 20,2010] Disponible en http://www.wordreference.com/definicion/.

[4] [l-linux] Trustix versus Ubuntu, Una mirada rápida a Trustix Secure Linux. [Online] [Citado: Febrero 10,2010] Disponible en http://swup.trustix.org/

[5] El APT Advanced Packaging Tool. [Online] [Citado: Febrero 05,2010] Disponible en http://www.codigolibre.org/index.php?option=com_content&view=article&id=3166%3Ael-3166&Itemid=54

[6] ClickOnce Deployment & Plataforma .Net. [Online] [Citado: Febrero 15,2010] Disponible en http://gahurtado.blogspot.com/2007/03/definiciones.html

[7] Ídem [6] Online.

[8] Ventajas Click Once. [Online] [Citado: Febrero 16,2010] Disponible en http://quomon.es/question-Qu%C3%A9-ClickOnce-ventajas-utilizarlo-1917.aspx

[9] Desventajas que tiene ClickOnce. [Online] [Citado: Febrero 16,2010] Disponible en http://www.forosdelweb.com/f29/problemas-clickonce-564854/

[10] Instalación de Click Once. [Online] [Citado: Febrero 17,2010] Disponible en http://www.eggheadcafe.com/software/aspnet/32368720/insatalacion-con-clickonc.aspx

[11] Introducción al Lenguaje C#. [Online] [Citado: Enero 21,2010] http://msdn.microsoft.com/es-es/library/z1zx9t92(VS.80).aspx

[12] Microsoft Corporation. MSDN Microsoft Developer Network. Visual Studio Developer Center. [En línea] MIcrosoft Corporation. [Citado: 21, 2010.] http://msdn.microsoft.com/en-gb/vstudio/bb265237.aspx

[13] Servicios en C# .Net. [Online] [Citado: Febrero 16,2010] Disponible en http://www.locualo.net/programacion/servicios-net/00000025.aspx

[14] Tecnología XML. [Online] [Citado: Enero 15,2010] Disponible en http://www.sparxsystems.com.ar/EAUserGuide/ea.html

[15] Gómez Velázquez, Karel Ing., Collada de la Rosa, Ricardo Ing. *Documento de Arquitectura de Software.2009*

[16] Programa C# sin visual Studio. [Online] [Citado: Enero 25,2010] Disponible en http://www.taringa.net/posts/downloads/2126623/Programa-C

[17] Guía de Usuario de Enterprise Architect. [Online] [Citado: Enero 25,2010] Disponible en http://www.sparxsystems.com.ar/EAUserGuide/ea.html

[18] Modelo de Dominio. [Online] [Citado: Febrero 01,2010] Disponible en http://iie.fing.edu.uy/ense/asign/desasoft/practico/hoja8/ejemplos_clase2.pdf

[19] Ing. Karel Gómez Velázquez, Ing. Ricardo Collada de la Rosa. *Documento de Arquitectura de Software.2009* Disponible en http://www.mono-project.com/Coding_Guidelines

BIBLIOGRAFÍA

1- ClickOnce Deployment & Plataforma .Net. [Online] [Citado: Febrero 15,2010] Disponible en http://gahurtado.blogspot.com/2007/03/definiciones.html

2- Conceptos básicos de Computación. [Online] [Citado: Enero 20,2010] Disponible en http://diccionario.babylon.com/aplicación informática

3- Conceptos básicos de Computación. [Online] [Citado: Enero 20,2010] Disponible en http://www.xuletas.es/ficha/conceptos-basicos-de-computacion/

4- Desventajas que tiene ClickOnce. [Online] [Citado: Febrero 16,2010] Disponible en http://www.forosdelweb.com/f29/problemas-clickonce-564854/

5- Diccionario de la lengua española. [Online] [Citado: Enero 20,2010] Disponible en http://www.wordreference.com/definicion/.

6- El APT Advanced Packaging Tool. [Online] [Citado: Febrero 05,2010] Disponible en http://www.codigolibre.org/index.php?option=com_content&view=article&id=3166%3Ael-3166&Itemid=54

7- Fuentes Suárez, Mainoldis y Ramos Medina, Alain; *Propuesta de arquitectura Sistemas de gestión Hospitalaria*; [UCI] 2007.

8- Gómez Velázquez, Karel Ing., Collada de la Rosa, Ricardo Ing. *Documento de Arquitectura de Software.2009*

9- Guía de Usuario de Enterprise Architect. [Online] [Citado: Enero 05,2010] Disponible en http://www.sparxsystems.com.ar/EAUserGuide/ea.html

10- [l-linux] Trustix versus Ubuntu, Una mirada rápida a Trustix Secure Linux. [Online] [Citado: Febrero 10,2010] Disponible en http://swup.trustix.org/

11- Instalación de Click Once. [Online] [Citado: Febrero 17,2010] Disponible en http://www.eggheadcafe.com/software/aspnet/32368720/insatalacion-con-clickonc.aspx

12- Introducción al Lenguaje C#. [Online] [Citado: Enero 21,2010] http://msdn.microsoft.com/es-es/library/z1zx9t92(VS.80).aspx

13- Larman, C. “UML y patrones” Tomo I Capítulos 18, Páginas 185-215

14- Legón García, Adonis César Ing.; *Servicio y componentes para la actualización automática de aplicaciones y recursos*; [UCI] 2009.

15- Libros de Computación [Online] [Citado: Abril 16,2010] Disponible en http://www.librosdeluz.net/como-crear-depositos-yum-en-linux-joel-barrios-duenas-libro-gratis/

16- Microsoft Corporation. MSDN Microsoft Developer Network. Visual Studio Developer Center. [En línea] MIcrosoft Corporation. [Citado: 21, 2010.] http://msdn.microsoft.com/en-gb/vstudio/bb265237.aspx

17- Microsoft Corporation. (s.f.). CSharp-Online.NET. Recuperado el 13 de 02 de 2010, de http://es.csharp-online.net/C_sharp_NET_/_Cap%C3%ADtulo_1

18- Modelo de Dominio. [Online] [Citado: Febrero 01,2010] Disponible en http://iie.fing.edu.uy/ense/asign/desasoft/practico/hoja8/ejemplos_clase2.pdf

19- Praag, J. v. (2006). AppGet. Recuperado el 30 de 01 de 2010, de http://www.app-get.com/whatisappget/

20- Programa C# sin visual Studio. [Online] [Citado: Enero 25,2010] Disponible en http://www.taringa.net/posts/downloads/2126623/Programa-C

21- Servicios en C# .Net. [Online] [Citado: Febrero 16,2010] Disponible en http://www.locualo.net/programacion/servicios-net/00000025.aspx

22- SIGHO. [En línea] [Citado el: 17 de febrero de 2010.] http://www.sigho.salud.gob.mx/descargas/pdf/manuales/actualizador_sighov3.3.0.pdf.

23- Sistema Gestión de paquetes [Online] [Citado: Abril 10,2010] Disponible en http://www.codigolibre.org/index.php?option=com_content&view=article&id=5344%3Asistema-de-gestion-de-paquetes&catid=36%3Agnu-cat&Itemid=53&showall=1

24- Tecnología XML. [Online] [Citado: Enero 15,2010] Disponible en http://www.sparxsystems.com.ar/EAUserGuide/ea.html

25- Uso Básico de APT [Online] [Citado: Abril 18,2010] Disponible en http://ubuntupatagonia.blogspot.com/2009/07/uso-basico-de-apt-advanced-packaging.html

26- Valerio, Adrián Anacleto. [En línea] [Citado el: 11 de Enero de 2010.] http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=15

27- Ventajas Click Once. [Online] [Citado: Febrero 16,2010] Disponible en http://quomon.es/question-Qu%C3%A9-ClickOnce-ventajas-utilizarlo-1917.aspx

ANEXOS

1. Descripción de casos de uso

Descripción del caso de uso: Modificar perfil

Objetivo	El objetivo que persigue con este CU es que luego de haber creado el perfil del producto para que se pueda llevar a cabo la nueva actualización este pueda ser modificado.	
Actores	Proveedor de actualizaciones: (Inicia) modificar perfil del producto.	
Resumen	El caso de uso inicia cuando el PA accede a la opción de modificar perfil, el sistema muestra los datos del perfil y brinda la posibilidad de cambiar sus valores ya sea introduciendo nuevos o seleccionando diferentes, el actor modifica los datos que necesita, el sistema actualiza los datos del perfil, el caso de uso termina.	
Complejidad	Baja	
Prioridad	Secundario	
Precondiciones	El perfil del producto ha sido creado.	
Postcondiciones	Se modificó el perfil.	
Flujo de eventos		
Flujo básico Crear perfil		
	Actor	**Sistema**
1.	El caso de uso inicia cuando el PA selecciona una el perfil y accede a la opción Modificar datos del perfil.	
2.		Muestra los datos: • Nombre de la solución • Versión • Licencia Seleccionar: • Plataforma • Velocidad mínima de CPU • RAM • Mínimo de video

		• Disco Brinda la posibilidad de cambiar sus valores ya sea introduciendo nuevos o seleccionando diferentes Y permite: Aceptar las modificaciones.
3.	Modifica los datos que necesita y selecciona la opción de aceptar las modificaciones.	El sistema muestra las especificaciones del nuevo perfil. Y muestra un mensaje de información ¨se ha actualizado los datos del perfil¨

Relaciones	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales	1,2,5,6	
Asuntos pendientes	Permitir que al crear el perfil y modificar el mismo se gestiones más especificidades del producto.	

Descripción del caso de uso: Modificar actualización

Objetivo	El objetivo es poder modificar la actualización de un producto luego de haber sido creada.
Actores	Proveedor de actualizaciones: (Inicia) modificar la actualización.
Resumen	El caso de uso inicia cuando el PA accede a la opción de modificar actualización, el sistema muestra los datos de la actualización y brinda la posibilidad de cambiar sus valores ya sea introduciendo nuevos o seleccionando diferentes, el actor modifica los datos que necesita, el sistema actualiza los datos de la actualización, el caso de uso termina.
Complejidad	Baja
Prioridad	Secundario
Precondiciones	Existe la actualización del producto.
Postcondiciones	Se modificó la actualización.
Flujo de eventos	
Flujo básico modificar actualización	

	Actor	**Sistema**
1.	El caso de uso inicia cuando el PA selecciona una versión y accede a la opción Modificar datos	

	de actualización.	
2.		Muestra los datos: • Nombre del proceso • Servicios • Clave • Valor • Dato • Nombre Seleccionar: • Archivo • Tipo • Evento • Acción Brinda la posibilidad de cambiar sus valores ya sea introduciendo nuevos o seleccionando diferentes Y permite: • Aceptar las modificaciones.
3.	Modifica los datos que necesita y selecciona la opción de aceptar las modificaciones.	
4.		El CU termina cuando el sistema manda un mensaje ¨se ha actualizado los datos de la actualización¨.

Relaciones	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales	1,2,3,4,5	
Asuntos pendientes	La actualización a de ser publicada.	

Descripción del caso de uso: Descripción de actualización

Objetivo	El objetivo es poder crear una descripción por cada actualización de un producto luego de haber sido creada, para especificar lo que esta corrige o añade.
Actores	Proveedor de actualizaciones: (Inicia) descripción de actualización.
Resumen	El caso de uso inicia cuando el PA procede a crear la nueva versión del producto, y desea crear su descripción, el caso de uso finaliza cuando el

	PA crea la descripción.	
Complejidad	Baja	
Prioridad	Auxiliar	
Precondiciones	Existe la actualización del producto.	
Postcondiciones	Se creó la descripción de la actualización.	
Flujo de eventos		
Flujo básico descripción de actualización		
	Actor	**Sistema**
1.	El caso de uso inicia cuando el PA selecciona la opción de crear la descripción de actualización.	
2.		El sistema brinda la posibilidad de llenar la descripción del perfil.
3.	Introduce en el sistema todas las especificidades de la nueva actualización.	
4.		El CU termina cuando el sistema guarda la descripción para ser utilizada por el usuario.
Relaciones	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales	1,2,3,4,5	
Asuntos pendientes	La actualización a de ser publicada.	

2. Diagramas de secuencia

CUS Gestionar producto

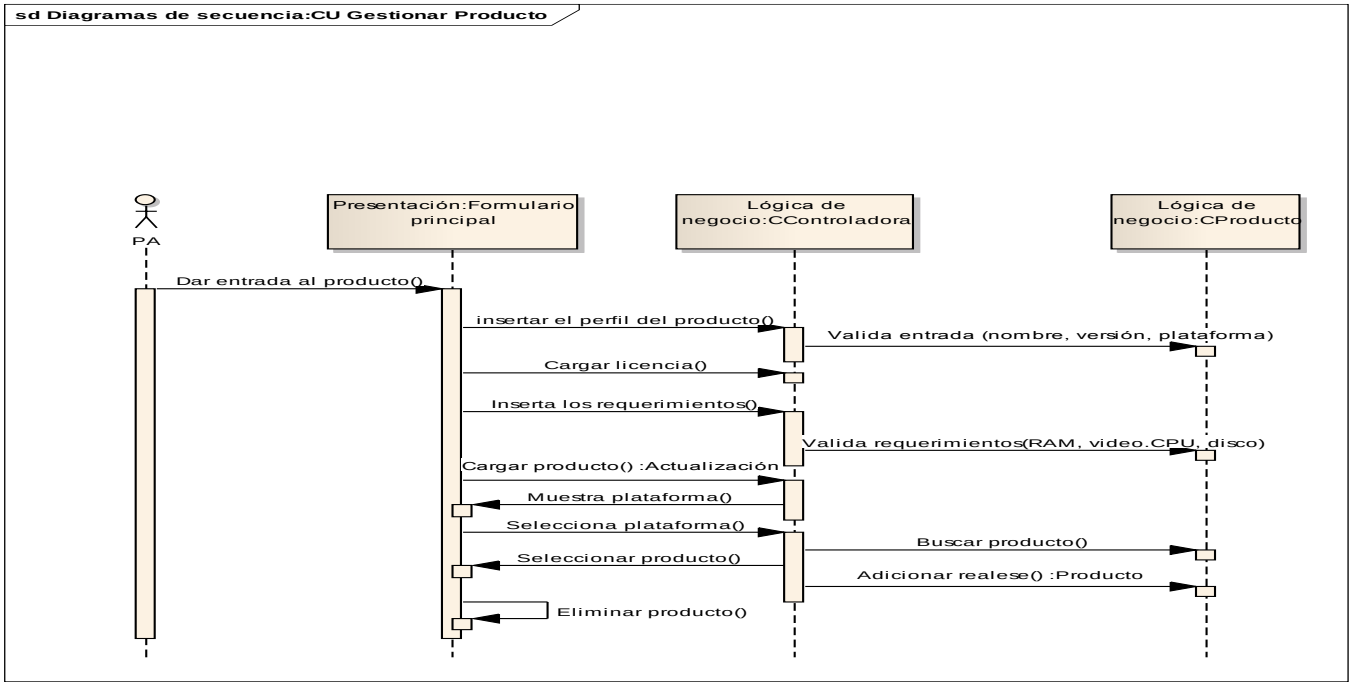

Fig. 38 Diagrama de secuencia CUS Gestionar producto

CUS Gestionar perfil

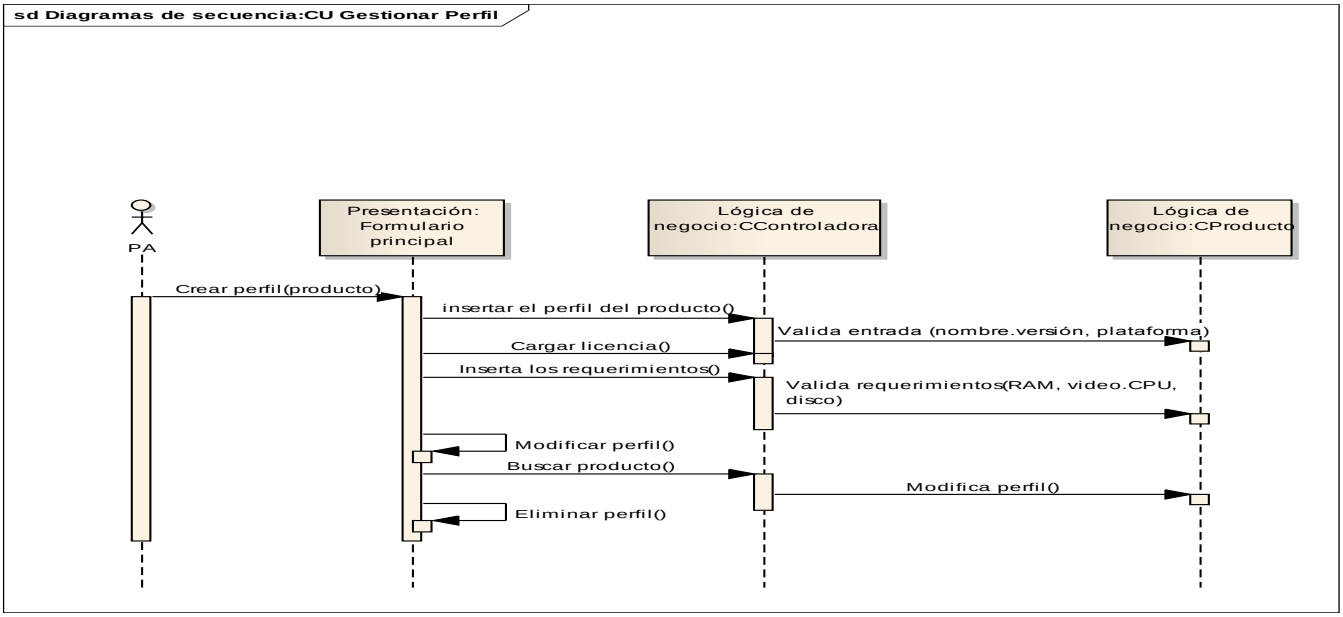

Fig. 49 Diagrama de secuencia CUS Gestionar perfil

CUS Gestionar Publicación

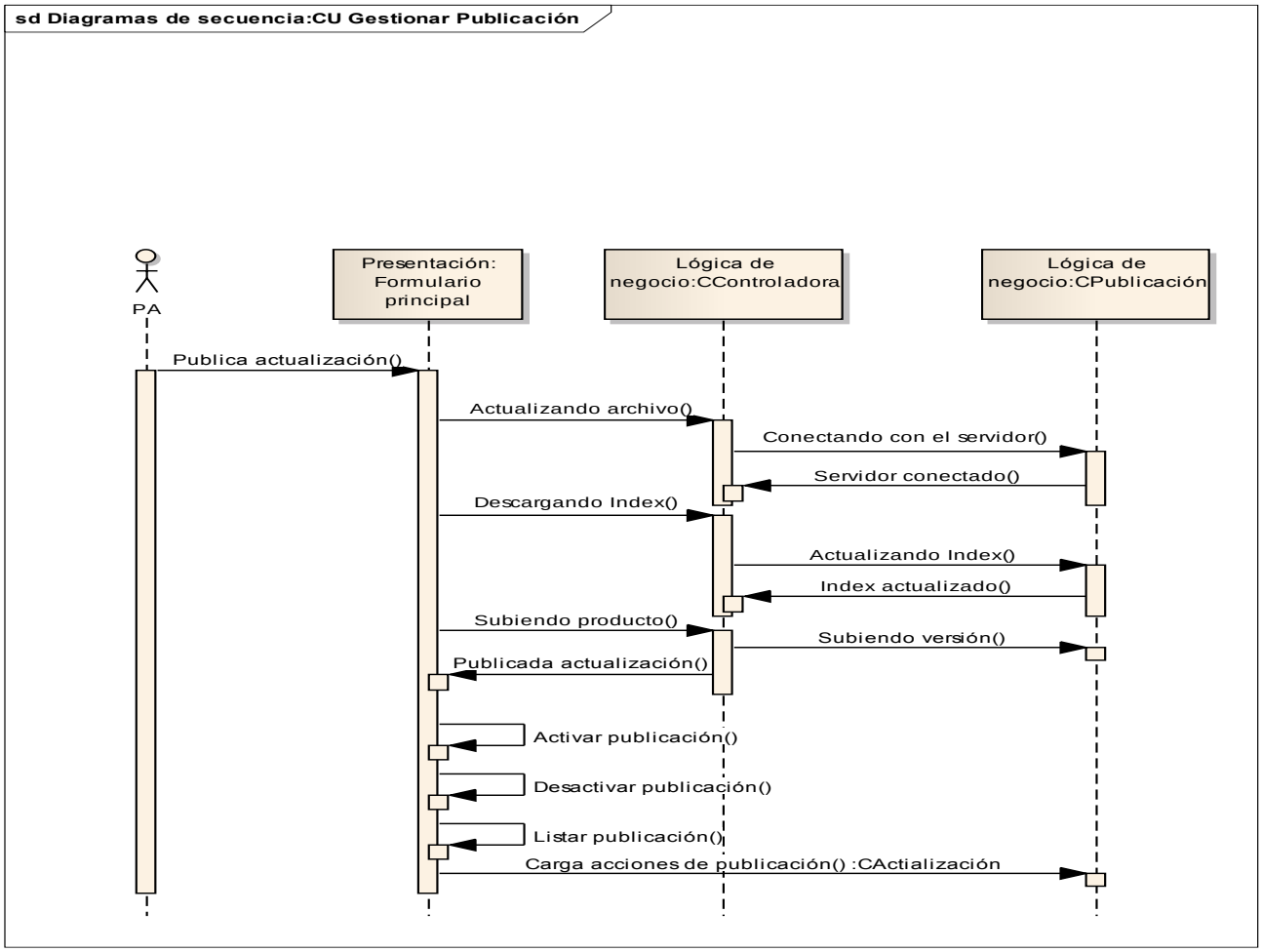

Fig. 20 Diagrama de secuencia CUS Gestionar publicación

CUS Gestionar Actualización

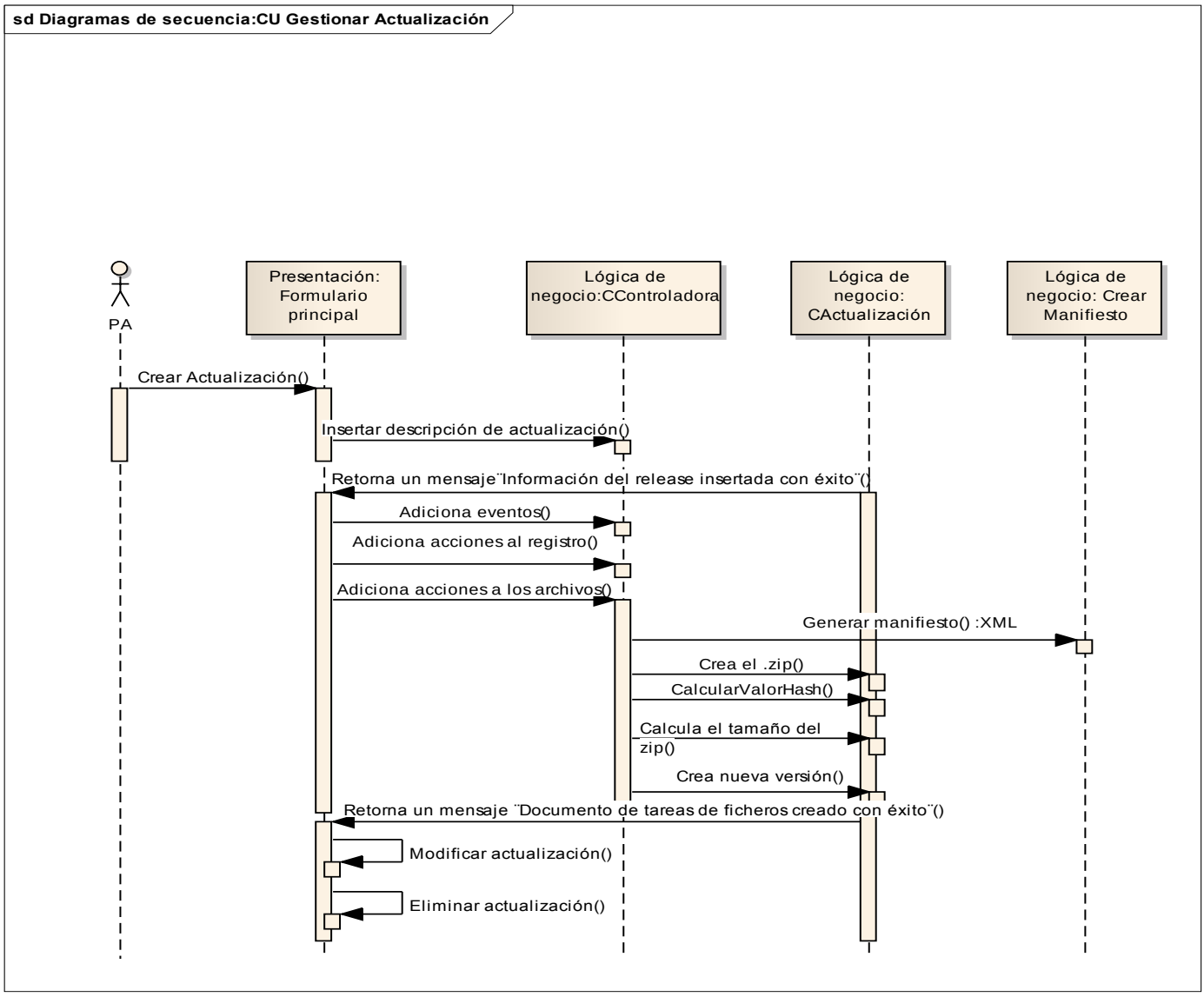

Fig. 21 Diagrama de secuencia CUS Gestionar actualización

GLOSARIO DE TÉRMINOS

1- Advanced Packaging Tool: Herramienta informática de gestión de paquetes creada por Debian.

2- Clases: Conjunto de objetos que comparten atributos, operaciones, relaciones y semántica, las mismas representan los conceptos fundamentales del sistema.

3- Diagrama de Clases: Representación de los conceptos de importancia en el área de la aplicación, así como de las relaciones entre estos.

4- ECMA-334: Estándar internacional que especifica la representación y semántica de programas escritos en C# así como la sintaxis y restricciones de este lenguaje.

5- Ecma International: Organización internacional que se basa en membrecías de estándares para la comunicación y la información.

6- Extensible Markup Language (XML): Un lenguaje de marcado extensible que puede usarse para almacenar datos en un formato estructurado, basado en texto y definido por el usuario.

7- Fichero ZIP: Extensión de los ficheros comprimidos con el algoritmo ZIP.

8- Librería: Conjuntos de código ya realizado que se puede reutilizar en los programas y que ahorran mucho esfuerzo en la programación.

9- Manifiesto: La clase manifiesto contiene un documento XML con el listado de la información de todos los recursos que el Actualizador necesita para actualizar la aplicación con las acciones que este debe realizar y pasos que llevará a cabo.

10- Paquete de Actualización: Serie de ficheros que juntos forman una versión de un software y están asociados a acciones que se describen en un documento XML. Estos ficheros y el documento XML conforman el paquete de actualización.

11- Parches: En informática, consta de cambios que se aplican a un programa, para corregir errores, agregarle funcionalidad o actualizarlo.