República de Cuba

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título

Sistema de Actualización para el Sistema de Gestión de Emergencias de la Seguridad Ciudadana

Autores:

Yamila Díaz Suárez

Carlos Dagnier Saavedra Ferras

Tutor:

Lic. Adrian Maranje Agramonte

"Año 54 de la Revolución"

La Habana, Cuba, Junio, 2012.

Declaración de Autoría

Por este medio declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2012.

Yamila Díaz Suárez: _____

Firma del Autor

Carlos Dagnier Saavedra Ferras: _____

Firma del Autor

Lic. Adrian Maranje Agramonte: _____

Firma del Tutor

Dedicatoria

Yamila

Dedico este trabajo de diploma a toda mi familia que ha sido siempre una guía firme en mi formación como persona y como profesional. Y en especial:

A mi mamá y a mi tía Milagros por darme tanto amor y apoyo toda mi vida, las quiero mucho.

A todos mis amigos que de una forma u otra han estado siempre ahí para apoyarme.

Carlos

A mis padres, porque creyeron en mí y me sacaron adelante, dándome ejemplos dignos de superación y entrega.

A mis amigos Ángel, Ismani, Tony y Alejandro, por estar a mi lado siempre y darme su amistad incondicional.

A mi tutor por su apoyo, comprensión, y sus consejos en los momentos difíciles.

A mi novia Osmara por toda su cooperación durante todo este tiempo.

A mis profesores Manfred, Javier, Rammel que me han dado toda su ayuda.

Espero no defraudarlos y contar siempre con su valioso apoyo, sincero e incondicional.

Agradecimientos

Yamila

A mi mamá y mi tía, mis dos ángeles de la guarda, por cuidarme y comprenderme siempre.

A mi papá por ser mi ejemplo de responsabilidad y consagración.

A mi tíos Nardo, Robel y Sucel por siempre estar ahí cuando los necesito.

A mi hermano, que a pesar de nuestras discrepancias lo quiero muchísimo.

A mis abuelos Nena, Pepa, Mateo que aunque no estén, siempre van a ocupar un pedacito en mi corazón.

A mis amigas de la UCI: Karina, Yuliani y Yolanda y los varones del 16201 por ser inseparables en estos últimos años.

A mis amigas de Holguín: Arianna, Tahimí, Darly e Ismary que a pesar de la distancia, nuestra amistad ha perdurado a través de los años y ocupan un lugar muy especial en mi corazón.

A mi Tommy por darme tanto apoyo en esta etapa que llevamos juntos. Te quiero mucho nene.

A todas las personas que me ayudaron incondicionalmente durante mi misión en la hermana República Bolivariana de Venezuela: Mirka, Mabel, Ana María, Ada, Diana y Yisel.

A Leandro y a su familia por haberme brindado mucho apoyo durante mi estancia en la UCI.

A todos mis compañeros del proyecto 171 por asesorarme y brindarme todo su apoyo en especial a Manfred, a Rammel y a mi querido tutor Adrián.

A mi compañero de tesis Carlos por soportarme en toda esta larga pero fructífera travesía.

A todo el que hizo posible que mi sueño se materializara y hoy haya terminado satisfactoriamente este trabajo de diploma.

A la vida, a Dios y a mi virgencita por permitirme llegar a donde estoy y haber vivido muchos momentos de felicidad durante esta etapa universitaria.

A Fidel por ser el principal impulsor de este gran Proyecto que es la UCI.

Carlos

Quiero agradecer a todos por su colaboración desinteresada y para la elaboración de esta tesis.

A mis padres, por todo su amor y comprensión.

A mi novia por haber estado a mi lado brindándome amor y comprensión en esta etapa tan importante de mi vida.

En especial a los profesores Rammel, Manfred y Javier que de una forma u otra me dieron toda la ayuda posible.

A mi tutor Adrián por servirme de guía.

A todas aquellas personas que me extendieron la mano y me apoyaron en mi investigación.

Resumen

La presente investigación se realiza con el objetivo de desarrollar un sistema capaz de llevar a cabo las actualizaciones automáticas de las aplicaciones que componen el Sistema de Gestión de Emergencias de la Seguridad Ciudadana (SIGESC).

Se definió un mecanismo de actualización que describe las funcionalidades que el sistema de actualizaciones automáticas debe permitir. Durante el desarrollo de este sistema, que cuenta con dos servicios de Windows y dos aplicaciones de administración para cada servicio, se utilizó como metodología de desarrollo de software el Proceso Unificado de Desarrollo, empleando la herramienta Visual Paradigm y el Lenguaje Unificado de Modelado para el modelado del sistema. Se definió una arquitectura híbrida para la construcción de la solución, donde se combinan lo mejor de la arquitectura de pares y el modelo cliente-servidor. Con el objetivo de estructurar la implementación se trabajó con el estilo arquitectónico tres capas reflejadas a través de las capas Presentación, Negocio y Acceso a Datos. Además se usó C# como lenguaje de programación sobre la plataforma .Net utilizando como entorno de desarrollo el Visual Studio 2010.

La solución es capaz de publicar, descargar y ejecutar varias actualizaciones, sin que esto interfiera en la calidad de la actualización. El sistema obtenido en esta investigación constituye un impacto positivo sobre el proceso de actualización de las aplicaciones que componen el SIGESC.

Palabras claves: Actualizaciones automáticas, Mecanismo de actualización, Sistema

Índice

Introducción

La evolución de la informática y las telecomunicaciones en las últimas dos décadas ha colocado a los sistemas informáticos en un rol sobresaliente dentro de las instituciones. Con este desarrollo ha habido un gran auge en la industria del software, dotando a la humanidad de medios eficaces para lograr la expansión del conocimiento, la cultura y la sabiduría.

Históricamente, el software era una forma estática de tecnología. Se adquiría un programa, se cargaba en el equipo y se utilizaba el software "tal como estaba" hasta que aparecía la próxima versión. Sin embargo, ese modelo ya no es aplicable en su totalidad. Los desarrolladores de software en la actualidad comprenden el beneficio de producir y ofrecer actualizaciones de los productos informáticos (Norton, 2008), principalmente para optimizarlos en tiempo y ejecución, utilizando algoritmos mejorados, añadiendo funcionalidades y eliminando secciones obsoletas. Con estos objetivos se han desarrollado diversas soluciones informáticas para lograr actualizar un software o varios al mismo tiempo. Una de las soluciones más conocidas a nivel mundial es el servicio en línea Windows Update, que ofrece Microsoft para actualizar sus Sistemas Operativos y aplicaciones.

Los mecanismos de actualización, automáticos o no, han estado orientados a resolver los problemas de mantenimiento y soporte de los sistemas y aplicaciones, partiendo de modelos como el de cliente-servidor, para hacer un uso efectivo de las redes deinformación y evitar costosos traslados de personal calificado para cumplir estos objetivos. La distribución de actualizaciones a través de redes de servidores y clientes aporta un valor agregado a este tipo de solución, e incorpora los conceptos de administración centralizada y gestión distribuida, ampliamente utilizados en muchos servicios y protocolos de comunicación, aunque no así para procesos de actualización de aplicaciones de propósito general (Campo, 2010). Nuestro país, a pesar de contar con pocos recursos y estar limitado el uso del Internet a grandes escalas, no está ajeno a estos avances tecnológicos. Existen algunos sistemas que utilizan este mecanismo para conservarse renovados, tales como el antivirus SAV (Segurmática Antivirus).

El Centro de Informatización de la Seguridad Ciudadana (ISEC), radicado en la Universidad de las Ciencias Informáticas (UCI), es el encargado de desarrollar productos, servicios y soluciones informáticas para la optimización del trabajo y mejoramiento de la calidad de la seguridad ciudadana. Una de las soluciones que se desarrollan es el Sistema de Gestión de Emergencias de Seguridad Ciudadana (SIGESC) que automatiza los procesos que se realizan dentro de los Centros de Atención de Emergencias 171 en la República Bolivariana de Venezuela, diseñada para funcionar sobre el sistema operativo Windows XP y desarrollada con la plataforma .Net. El sistema está conformado por nueve

aplicaciones de escritorio, una aplicación web, un servidor de sincronización y un servidor de base de datos, siendo clasificado como una solución distribuida.

El hecho de que el equipo de desarrollo se encuentra en la UCI y el SIGESC esté desplegado en varios estados de la República Bolivariana de Venezuela representa una dificultad para el proceso de actualización de las aplicaciones que lo componen. Esta dificultad está sustentada en la carencia de un mecanismo económico, rápido y confiable para hacer llegar dichas actualizaciones hasta los lugares donde está instalado el sistema informático. Las actualizaciones son desplegadas manualmente por un equipo de soporte que en ocasiones comete errores de omisión de pasos u otros durante el proceso, provocando que el sistema quede en estado de versión incoherente y afectando de esta manera el correcto desempeño del mismo.

Atendiendo a la situación problemática anteriormente descrita el problema que se impone resolver consiste en: ¿Cómo garantizar que el SIGESC disponga de las actualizaciones más recientes de forma rápida y confiable?

El **objeto de estudio** se enmarca en el proceso de actualización de sistemas informáticos y el **campo de acción** se centra en los sistemas de actualizaciones automáticas.

Para dar solución al problema antes mencionado se propone como **objetivo general:** Desarrollar un sistema informático para la actualización automática de las aplicaciones que componen el SIGESC.

De acuerdo con esta propuesta se derivan los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación.
2. Definir un mecanismo para la realización del proceso de actualización automático del SIGESC.
3. Implementar un sistema informático que responda al mecanismo de actualización propuesto.
4. Validar la solución propuesta.

Para cumplir con los objetivos anteriormente planteados y resolver la problemática que ocupa esta investigación se proponen las siguientes **tareas a resolver**:

1. Análisis de los requisitos que componen el mecanismo de actualización orientado a satisfacer las necesidades del SIGESC.
2. Realización de una investigación sobre las diferentes tecnologías existentes para la realización automática de tareas de actualización de sistemas informáticos.
3. Selección de las herramientas, metodologías y tendencias que se adapten a la situación.
4. Descripción de un mecanismo para la realización de las actualizaciones automáticas de las aplicaciones del SIGESC.
5. Obtención de los requisitos funcionales y no funcionales del sistema.
6. Definición de la arquitectura del sistema.

7. Definición, modelación y descripción de los elementos del diseño para la posterior construcción de la solución.
8. Implementación de un prototipo funcional que permita dar solución al problema planteado.
9. Definición del tipo de prueba a realizar al sistema propuesto.
10. Realización de los casos de prueba.

Una vez finalizada la investigación, la solución desarrollada permitirá:

1. Un proceso automatizado de actualización ajustado a las necesidades del SIGESC.
2. Evitar el retraso y los errores de numerosas actividades que puedan ser ejecutadas por el equipo de soporte, ya que garantizará la distribución de los paquetes de actualización y la instalación de los mismos de manera rápida y confiable.

Para el desarrollo de las tareas de la investigación se tienen en cuenta los siguientes métodos investigativos.

Analítico-Sintético. Se utiliza este método centrándose en el análisis de las teorías, documentos, permitiendo la selección de los elementos más importantes en cuanto a los sistemas de actualizaciones automáticas, de manera que permita elaborar correctamente la información.

Inductivo-Deductivo. Se utiliza este método con el objetivo de llegar a conocimientos generalizadores que permitan determinar qué procesos y mecanismos que permitirán obtener resultados eficientes para el sistema de actualización automática.

Histórico-lógico: Este método se utiliza con el objetivo de constatar teóricamente la evolución y características de los sistemas existentes de actualizaciones automáticas así como las metodologías, herramientas y tecnologías que se utilizarán para el desarrollo del trabajo.

El presente trabajo se estructura en tres capítulos definidos de la siguiente forma:

Capítulo 1 Fundamentación Teórica: Se analizan los requisitos necesarios que debe componer un mecanismo de actualización para el SIGESC. Se hace un estudio de distintos sistemas de actualizaciones automáticas en el mundo y en la universidad teniendo en cuenta que cumplan con los requisitos analizados. Además se define la metodología, lenguaje, plataforma y herramientas a utilizar para la construcción del mecanismo de actualización.

Capítulo 2 Construcción de la solución: Se especifican las características que debe tener el mecanismo de actualización, se propone un flujo de actividades y se identifican las entidades que interactúan en dichas actividades. Se definen las características del sistema así como se hace un análisis del dominio de la aplicación, se describen las funcionalidades a automatizar para darle solución al problema. Se define el diseño del sistema siguiendo la metodología Proceso Unificado de Desarrollo y se elaboran los diagramas

de clases del diseño. Además se definen el diagrama de componentes y de despliegue del sistema implementado.

Capítulo 3 Validación de la solución: Se define el método de prueba a utilizar y se diseñan los casos de pruebas correspondientes a los casos de uso del sistema con el objetivo de probar funcionalmente la solución. Se realizan además pruebas de rendimiento del sistema para demostrar que se dio solución al problema planteado.

Capítulo I: Fundamentación Teórica

1.1 Introducción

El mundo digital actual es muy dinámico y todos sus procesos son variables y se perfeccionan diariamente, así mismo sucede con el software el cual requiere de herramientas automáticas para lograr mantenerse actualizado. En el presente capítulo es trascendente el análisis de los requisitos que componen un mecanismo de actualización para el SIGESC con el objetivo de garantizar un producto acorde con las necesidades actuales. Es relevante también el estudio de las características de algunos sistemas de actualizaciones automáticas existentes, basándose en el cumplimento de estos requisitos, con el objetivo de encontrar elementos que puedan ser analizados para la determinación de la propuesta. Posterior al estudio de estos temas es necesario definir la metodología de desarrollo a utilizar, lenguaje de modelado, lenguaje de programación, así como las tecnologías y herramientas que servirán para la construcción de la solución.

1.2 Requisitos para el mecanismo de actualización del SIGESC

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 (Ajmani, 2004). Una actualización del software contiene alteraciones apropiadas de la funcionalidad y de la configuración, esta actividad constituye para muchas organizaciones un proceso en el cual deben existir la menor cantidad de errores para que se logre una actualización exitosa de los sistemas informáticos.

El SIGESC, como sistema distribuido, es un sistema informático que se encuentra dividido en componentes de software que se localizan en diferentes ordenadores. Dichos componentes se comunican mediante la red y actúan coordinadamente con el objetivo de satisfacer las necesidades para los que se crearon. Actualmente el mecanismo de actualización que posee el SIGESC no se encuentra debidamente definido y constituye una actividad que demanda dinero, tiempo y cierta experiencia por parte del personal encargado de realizarla.

Las actualizaciones del SIGESC se distribuyen en paquetes de Windows Installer que son creados por un equipo de desarrollo localizado en la UCI y estos son desplegados e instalados manualmente por un equipo de soporte localizado en una geografía distante, careciendo de un método automatizado de transferencia de archivos para hacer llegar en tiempo dichas actualizaciones a los centros de atención de emergencias donde se encuentra instalado el sistema. Otro de los inconvenientes de realizar esta tarea lo constituye el hecho de trasladar a dicho personal hasta el lugar donde se desplegarán las actualizaciones, incurriendo en gastos por conceptos de transportación, alojamiento, comida y otros. El personal

encargado de realizar las instalaciones y actualizaciones muchas veces no cuenta con la experiencia necesaria para realizar estas actividades, lo que influye directamente en que se puedan cometer diversos errores al actualizar las aplicaciones y estas funcionen de forma incoherente.

Los problemas antes descritos demuestran que el proceso de actualización que posee actualmente el SIGESC carece de rapidez y confiabilidad, siendo fundamental que se analicen los requisitos que debe mejorar el mecanismo de actualización existente, con el objetivo de que la solución que se proponga en esta investigación mitigue estas deficiencias.

1.2.1 Instalación de paquetes de Windows Installer

La instalación y actualización son procesos necesarios para que un sistema informático logre ser utilizado y se explote el potencial de las nuevas versiones. Para agilizar la realización de estas tareas se han desarrollado tecnologías que permiten administrar estos procesos, es el caso de Windows Installer.

Windows Installer es un servicio de instalación y configuración que se distribuye como parte del sistema operativo Microsoft Windows, logrando que todos los equipos mantengan una base de datos de información sobre cada aplicación que se instala, incluidos los archivos, las claves del registro y los componentes (Microsoft Corporation, 2007).

Los paquetes de Windows Installer se distribuyen a través de archivos de tipo MSI[1] y archivos MSP[2], el SIGESC solo se vale de paquetes MSI para la distribución de los instaladores y de las actualizaciones de las aplicaciones, y estos a su vez se crean aunque se realicen cambios pequeños dentro de las aplicaciones. Por tal motivo la actualización de una aplicación del SIGESC se realiza desinstalando la versión actual e instalando el nuevo paquete.

Los beneficios que trae el manejo de archivos MSI y MSP, para un sistema como el SIGESC, lograrían un paso de avance en el mecanismo de actualización al permitir distribuir las actualizaciones como parches para solución de problemas específicos logrando que la aplicación se actualice sin tener que desinstalarla e instalarla completamente.

1.2.2 Estado coherente del sistema informático

La coherencia dentro los sistemas informáticos es hoy en día un aspecto tomado en cuenta por los equipos de desarrollo, aunque en la práctica es difícil de mantener debido a la cantidad de información que un software maneja, aun más cuando se trata de una solución distribuida como el SIGESC, donde debe existir una relación o lógica en el funcionamiento de todas las aplicaciones que lo conforman.

[1]MSI: MicrosoftInstaller, Paquete de Windows Installer

[2]MSP: Microsoft InstallerPatch, Revisión de Windows Installer o Paquete de actualización.

Después de ejecutar la instalación y/o actualización sobre un sistema informático es importante que no existan fallos en su funcionamiento, en la actualidad se han diseñado software que corrigen estos problemas, ejemplo de estas aplicaciones es el software Imago (Dumitras, 2009) que utiliza la compañía Wikipedia para actualizar su sistema web, pero de manera general no son realmente adaptables para ser utilizadas en aplicaciones de escritorio como las del SIGESC.

En el epígrafe anterior se analizó el uso de paquetes de Windows Installer para el mecanismo de actualización, sin embargo la instalación de estos paquetes sin concebir un control inclinado a mantener sincronizadas las aplicaciones que se instalan en cada estación de trabajo ocasionaría un incorrecto desempeño del sistema distribuido.

Uno de los factores que afectan la coherencia de las aplicaciones que componen el SIGESC está dado principalmente por la inconsistencia entre las aplicaciones de un mismo tipo instaladas en un centro de atención de emergencias, dado que se instalan distintas versiones de una misma aplicación que provocan cambios en la lógica de funcionamiento del sistema.

1.2.3 Disponibilidad del mecanismo

En escenarios de la informática, la disponibilidad hace referencia a la probabilidad de que un servicio funcione adecuadamente en cualquier momento (Jeff, 2008). Este término representa la continuidad de acceso a los elementos de información almacenados y procesados en un sistema informático. Basándose en este principio, las herramientas de seguridad informática deben reforzar la permanencia del sistema informático, en condiciones de actividad adecuadas para que los usuarios accedan a los datos con la frecuencia y dedicación que requieran. La disponibilidad es importante en sistemas informáticos cuyo compromiso con el usuario es prestar servicio permanente.

En el mecanismo de actualización actual del SIGESC, los problemas que ocasiona la falta de disponibilidad están dados debido a que el desempeño de las actividades de despliegue son realizadas por un pequeño equipo de soporte que no puede proporcionar de forma paralela la instalación de las actualizaciones hacia todos los centros de emergencias, impidiendo que se puedan utilizar las actualizaciones inmediatamente en cada centro. Esta demora de las actividades de mantenimiento y soporte también ocasiona que se liberen actualizaciones al mismo tiempo que se despliegan las que anteriormente fueron liberadas.

En general los problemas de disponibilidad de los sistemas informáticos se han visto solucionados a través de la implementación de aplicaciones que comienzan a funcionar de manera automática tras el inicio del sistema operativo y continúan ejecutando sus tareas mientras no se apague la computadora o un usuario las detenga a través de las herramientas de administración. Un mecanismo de actualización que garantice coherencia entre sus aplicaciones debe también proporcionar disponibilidad y que todas las

tareas previstas se dispongan a través de aplicaciones que funcionen mientras el sistema operativo esté activo y de esta manera lograr mantener actualizado el software en un tiempo aceptable después de liberada la nueva versión.

1.2.4 Transferencia de archivos

La distribución de archivos es una tecnología muy variada y versátil ya que no se limita a un solo mecanismo pues las distintas características de las redes a nivel mundial conllevan a desarrollar nuevas técnicas de distribución. Un archivo o fichero puede definirse como un conjunto de información que se almacena en forma virtual para ser leído y/o accedido por medio de una computadora (Victoria, 2007).

Uno de los principales problemas del mecanismo de actualización del SIGESC lo constituye la distribución manual de los ficheros de actualización, debido a las desventajas económicas que ocasiona pues es costoso transportar y brindar alojamiento y comida a un equipo de soporte, cada vez que se libere una nueva actualización.

En la medida en que los sistemas informáticos han evolucionado también han surgido nuevas formas de distribuir archivos que logran realizarse sin conducir a grandes gastos económicos. Por lo tanto es necesario el estudio de las principales tecnologías que existen para la transferencia de archivos por la red con el objetivo de analizar las ventajas y desventajas del uso de las mismas.

1.2.4.1 Protocolo de transferencia de Hipertexto

El protocolo de transferencia de hipertexto (HTTP[3]) se utiliza bajo la arquitectura cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. Está soportado sobre los servicios de conexión TCP[4]/IP[5], su funcionamiento se basa en un proceso servidor escuchando en un puerto de comunicaciones TCP, y espera las solicitudes de conexión de los clientes. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libres de errores.

HTTP se basa en operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML[6], fichero multimedia o aplicación CGI[7])

[3]HTTP: Hypertext Transfer Protocol

[4]TCP: Transmission Control Protocol, Protocolo de Control de Trasmisión

[5]IP: Internet Protocol, Protocolo de Internet

[6]HTML: HyperText Markup Language, Lenguaje marcado de hipertexto

[7]CGI: Common Gateway Interface, Interfaz de entrada común

es conocido por su URL[8]. Es un protocolo sin manejo de estados pues tras cada respuesta el servidor cierra inmediatamente la conexión.

1.2.4.2 Servicio de Transferencia Inteligente en Segundo Plano

El Servicio de Transferencia Inteligente en Segundo Plano (BITS[9]) está dirigido a solucionar el problema de la transferencia de archivos entre un cliente y un servidor en sistemas operativos Windows, a diferencia de otros protocolos de transferencia utiliza solo el ancho de banda de red inactivo. Realiza las transferencias de forma asincrónica, así como también suspende las transferencias si el usuario cierra la sesión o la conexión de red se pierde. Soporta el protocolo HTTP y HTTPS[10] incluyendo la carga y descarga (Cabai, 2008).

BITS ofrece grandes beneficios a las tecnologías de Microsoft Windows por los siguientes aspectos:

- Puede transferir archivos en primer y segundo plano.
- Si el proceso de transferencia de archivos se interrumpe, BITS puede reanudarlo desde el punto exacto de la interrupción, en lugar de volver a transferir todo el archivo.
- Posee manejabilidad pues proporciona interfaces y métodos que se utilizan para administrar el trabajo de transferencia.
- Utiliza la API[11] de BITS para especificar las credenciales a utilizar para un proxy o petición al servidor de autenticación de usuarios, siendo así lo más seguro posible.
- Permite la descarga de intervalos de archivos de modo que un programa cambie el origen de transferencia de un archivo.
- Mejora el consumo de ancho de banda del cliente que descarga la actualización.

Sin embargo BITS también cuenta con las siguientes desventajas:

- BITS no soporta cargas y descargas de archivos de forma concurrente, permite la transferencia de un solo archivo a la vez.
- No permite proveer una estimación de cuánto tiempo toma realizar una descarga de un archivo basado en el ancho de banda y el tamaño del archivo.

[8] URL: Uniform Resource Locator, Localizador Uniforme de Recursos

[9] BITS: Background Intelligence Transfer Service

[10] HTTPS: Hypertext Transfer Security Protocol, Protocolo seguro de transferencia de hipertexto

[11] API: Application Programing Interface,Interfaz de programación de aplicaciones de Windows

1.2.4.3 Protocolo de Transferencia de Archivos

El Protocolo de Transferencia de Archivos (FTP[12]) se desarrolló para permitir las transferencias de archivos entre un cliente y un servidor. Un cliente FTP es una aplicación que se ejecuta en una computadora y se utiliza para cargar y descargar archivos desde un servidor. Para transferir los archivos en forma exitosa, el FTP requiere de dos conexiones entre cliente y servidor: una para comandos y respuestas, otra para la transferencia real de archivos. La transferencia de archivos puede producirse en ambas direcciones. El cliente puede descargar un archivo desde el servidor o el cliente puede cargar un archivo en el servidor (Cisco Networking Academy, 2011).

Las ventajas del FTP incluyen:

- Soporte para todos los tipos de cliente: la implementación estandarizada del protocolo significa que prácticamente cualquier cliente FTP puede utilizar el servidor FTP.
- Alto rendimiento y sencillez: el rendimiento y sencillez del protocolo lo hacen una opción conveniente para la transferencia de archivos a través de Internet.

Desventajas:

- Los datos e información de inicio de sesión se envían sin encriptación a través de la red. Esto tiene como resultado el descubrimiento de las cuentas o contraseñas de inicio de sesión. Esta información se puede utilizar por personas no autorizadas para acceder a otros sistemas (Microsoft, 2011).
- La disponibilidad del contenido depende de la disponibilidad del servidor, esto significa que si por algún motivo el servidor falla, el contenido ofrecido deja de estar disponible, ocasionando la interrupción total de la descarga (Pardo Tapia & Martínez Campos, 2010).
- Al existir un tráfico grande en la red se sobrecarga un solo punto llegando a colapsar el servicio.

1.2.4.5 Protocolo BitTorrent

BitTorrent es un protocolo de compartición de archivos que se utiliza bajo la arquitectura P2P[13] que implementa un sistema híbrido basado en redes dedicadas por cada archivo (o grupos de archivos) que se desee compartir, estando enfocado como una solución para archivos de gran tamaño (Pardo Tapia & Martínez Campos, 2010).

Una red BitTorrent está formada por:

- *Peers* (pares): Se denomina así a todos los usuarios que están en la red.
- *Leechers* (sanguijuelas): Se denomina así a todos los usuarios que están en la red descargando el archivo pero que todavía no tienen el archivo completo.

[12] FTP: File Transfer Protocol

[13] P2P: Peer to Peer, Igual a Igual, Punto a Punto o Arquitectura de pares

- *Seeds* (semillas): Son los usuarios de la red que poseen el archivo completo. Solo suben partes a los demás pares, pero no bajan nada.
- *Tracker* (rastreador): Es un servidor especial que contiene la información necesaria para que los pares se conecten unos con otros. Inicialmente es la única forma de localizar qué usuarios contienen el archivo que se quiere descargar.
- Archivo Torrent (torrente): Un archivo Torrent contiene metadatos que indican todos los archivos que pueden bajarse a través de él, nombres, tamaños, hashes de todas sus partes y la dirección del rastreador.

La red BitTorrent básicamente se caracteriza por dotar al cliente de un archivo Torrent obtenido desde un servidor Tracker, el cual asigna a cada cliente, "n" posibles destinatarios y de esta forma conectarse a ellos para lograr una descarga descentralizada (BitTorrent, 2012). BitTorrent manipula para lograr su funcionamiento el protocolo de transporte TCP. Y sus ventajas están dadas fundamentalmente por:

- Permite que quienes descargan, a la vez usen su capacidad para enviar el archivo a otros.
- Escalabilidad, dado que un nuevo participante no solo agrega demanda, sino que también aporta recursos de distribución (Calderón, 2006).
- Se considera un protocolo caracterizado por su robustez, ya que fallas en la red no producen el fin de la descarga, sino pausa en ella.
- Se puede utilizar con cualquier tipo de conexión.
- Tiene chequeos de errores por cada segmento del archivo que baja, por tanto hay más posibilidades de bajar el archivo correctamente.

Entre sus desventajas están:

- Archivos no deseados como virus, gusanos y troyanos pueden ingresar a los ordenadores como archivos Torrent.
- No es un protocolo totalmente distribuido pues los archivos Torrent tienen que ser distribuidos a través de canales externos.
- La gestión de ancho de banda no es fácil de usar, por lo general el tráfico de BitTorrent interfiere con el tráfico de otros usuarios, lo que frena las aplicaciones como la navegación web o correo electrónico, el uso de TCP como protocolo de transporte y sus algoritmos fijos de control de congestión limita las acciones posibles en este camino.

Sin embargo en un intento por mejorar el último problema planteado las empresas detrás del desarrollo de BitTorrent realizaron un rediseño al protocolo usando en lugar de TCP, el protocolo uTP[14], trabajando

[14] uTP: Micro Transport Protocol, Micro Protocolo de Transporte.

sobre el protocolo UDP[15] para transferencias de punto a punto. El protocolo uTP permite que su propio tráfico se ralentice cuando se detecta que la latencia de la red aumenta, haciéndolo más accesible a los sistemas que deben soportar grandes cargas. En consecuencia uTP haría las transferencias BitTorrent más livianas para la red, más eficientes y menos agresivas.

1.3 Sistemas y mecanismos de actualización automática

La acelerada evolución de las tecnologías de información, la creciente complejidad de los requerimientos de negocio, los continuos cambios en el contexto de desarrollo, entre otros aspectos ocasionan una rápida obsolescencia del software (García, 2008). En la mayoría de los casos un software no tiende a actualizarse por sí mismo sino que necesita de herramientas que permitan reemplazar su versión obsoleta. Un sistema de actualizaciones automáticas debe estar compuesto de al menos dos software, un servidor y un cliente para realizar la transmisión de los datos que se desean actualizar de forma automática, sin necesidad de que exista un personal capacitado atendiendo este proceso.

Existen actualmente soluciones particulares, que no solo se limitan a actualizar diferentes programas externos sino que también los instalan, el análisis de estas soluciones parte de los requisitos antes expuestos.

1.3.1 Windows Server Update Service (WSUS[16])

Windows Server Update Service automatiza el proceso de gestión de parches y actualizaciones de Microsoft Windows sin limitarse a cuestiones del sistema operativo sino que también gestiona parches de otros productos Microsoft, es una herramienta de dominio público. La infraestructura de WSUS permite que desde un servidor central se descarguen automáticamente los parches y actualizaciones para los usuarios en la organización, en lugar de hacerlo del sitio web Microsoft Windows Update[17](Ver **figura 1.1**). Este mecanismo ahorra ancho de banda de Internet, tiempo y espacio de almacenamiento debido a que las computadoras no necesitan conectarse individualmente a servidores externos a la organización, sino que se conectan a servidores locales.

[1515]UDP: User Datagram Protocol, Protocolo de Datagrama de Usuario.

[16]WSUS: Servicio de Actualización de Windows Server

[17]Windows Update es un servicio en línea creado por Microsoft con el objetivo de descargar actualizaciones adecuadas para Windows.

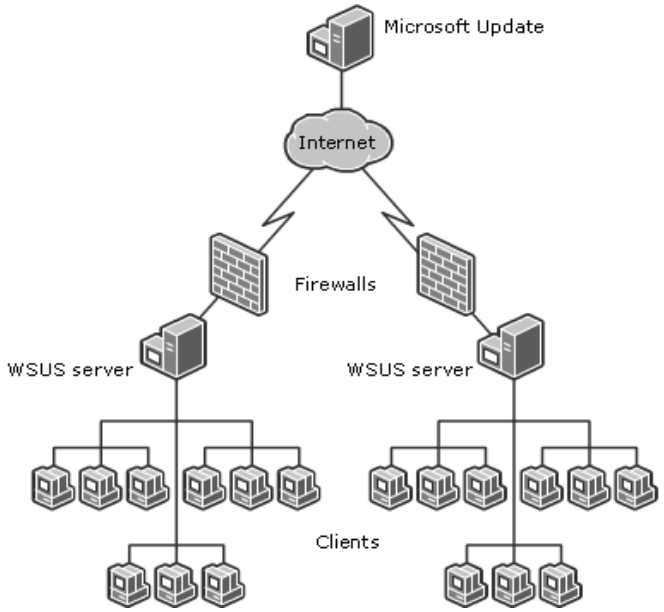

Figura 1.1- Despliegue de Windows Server Update Service tomado de (Elhajj, 2005)

WSUS se administra en el propio servidor mediante una consola de administración. Utiliza BITS para controlar el ancho de banda a utilizar para la transferencia de archivos y que la descarga de estos no afecte otras tareas dentro del sistema operativo. El servidor de WSUS manipula los metadatos u "objetos diccionario" que describen los paquetes disponibles y su aplicabilidad. Esta información se descarga en un archivo llamado Aucatalog1.cab. Los metadatos se almacenan en la base de datos de WSUS.

Esta herramienta implementa un mecanismo de actualización bastante efectivo pues manipula acciones personalizadas para instalación y la actualización, está disponible permanentemente haciendo uso de un protocolo flexible para las descargas, sin embargo está orientada generalmente a brindar servicio a sistemas monolíticos[18], no a soluciones distribuidas como el SIGESC donde es fundamental mantener la lógica en el funcionamiento de las distintas aplicaciones que se ejecutan en varias estaciones.

1.3.2 Microsoft Systems Center[19]

Microsoft System Center es una familia de soluciones para administración de soluciones en plataforma TI[20], que ayuda a planear, implantar, administrar, operar y optimizar un ambiente de TI (Microsoft, 2012). Microsoft System Center permite entre sus diversas funciones la automatización del proceso de instalación, actualización y mejoras del software, publicando varios productos con estos objetivos para

[18]Los sistemas monolíticos son aquellos que no tienen una estructura totalmente clara y definida, sino que se encuentran todos en un solo conjunto o programa (Pérez, 2004).

[19]Centro de Sistemas de Microsoft

[20]Tecnologías de la Información

brindar servicios en un entorno corporativo. Dos de los productos más recientes lo constituyen el System Center Essentials 2010 y el System Center Updates Publisher 2011 (SCUP).

System Center Essentials 2010

System Center Essentials es una aplicación de servidor para el control y la instalación de servicios de TI y el software de actualización a través de una red. Essentials automatiza y estandariza el proceso de instalación y actualización de software en servidores y clientes de una organización. Ofrece una forma integrada de saber no solamente qué aplicaciones están instaladas en un equipo, sino su nivel de utilización. Al integrarse plenamente con la plataforma Windows y con las tecnologías de gestión como Windows Management Instrumentation (WMI), Directorio Activo y Windows Installer Services, System Center Essentials 2010 ofrece una mayor capacidad de gestión para los clientes basados en Windows.

System Center Updates Publisher 2011

System Center Updates Publisher es una herramienta que permite a los proveedores independientes de software o a los desarrolladores de aplicaciones de negocios importar catálogos de actualizaciones de software, crear y modificar definiciones de actualizaciones y publicar información de actualizaciones en un servidor configurado (System Center, 2011). Updates Publisher ofrece la capacidad de publicar el contenido completo para actualizaciones de software, los metadatos, o simplemente los metadatos de las actualizaciones de software.

Estas soluciones utilizan BITS para la transferencia de archivos entre un cliente y un servidor y para la gestión del ancho de banda. Manipulan acciones personalizadas para la actualización y trabajan con paquetes MSI y MSP de una forma muy similar a WSUS, siendo un servicio que puede permanecer disponible las 24 horas, sin embargo es limitada la adquisición de estos productos analizados debido a que utilizan licencias privativas y solo facilitan el proceso de actualización a los sistemas para los cuales han sido definidos y adaptados, tales como los distintos sistemas operativos y aplicaciones de Microsoft.

1.3.3 Sistemas existentes en la UCI

La Universidad de las Ciencias Informáticas desarrolla en estos momentos un conjunto de proyectos que manejan la solución de los problemas de las actualizaciones automáticas de los productos que se desarrollan en los mismos, así estos brindan un mejor mantenimiento y soporte en los despliegues de software.

1.3.3.1 Servicio de Actualización Automática del SAIME[21]

Este producto está compuesto por un componente que permite la actualización automática de las aplicaciones del SAIME y del propio servicio de actualización, a partir de la publicación de nuevas versiones que se descargan directamente desde un servidor o mediante servidores intermediarios en los que se actualizan y publican las actualizaciones, permitiendo configurar una red de distribución de actualizaciones (Ver **Figura 1.2**). Actualmente funciona sobre protocolo HTTP pero es posible integrar otros protocolos para la descarga de los recursos que conforman la actualización tales como FTP y SFTP (Campo, 2010). Permite realizar actualizaciones de forma jerárquica, con posibilidades de integrarse a las aplicaciones como un componente o de funcionar de forma independiente como un servicio, y con la capacidad para ser extendidas tanto las operaciones que se ejecutan durante el proceso de actualización como el mecanismo para la descarga de los recursos que la conforman.

Este servicio de actualización posee varias características que son adaptables al SIGESC por ser una solución que permite la actualización de un sistema distribuido como el SAIME, sin embargo no contemplan en su mecanismo una solución para mantener un estado coherente de las aplicaciones que componen el sistema, no manejan la instalación de paquetes de Windows Installer y no utilizan un protocolo de transferencia de archivos efectivo dado que una sobrecarga en el servidor central ocasiona demoras en el proceso de actualización y hasta la interrupción definitiva de la descargas de ficheros.

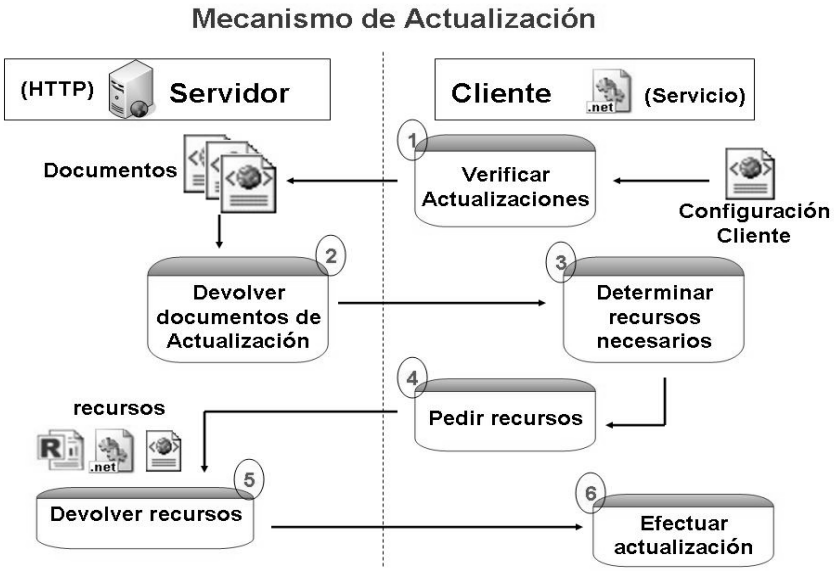

Figura 1.2- Diagrama general del proceso de actualización del SAIME tomado de (Campo, 2010)

[21]SAIME: Servicio Autónomo de Identificación Migración y Extranjería

1.3.3.2 Sistema de actualización automática para el Sistema de Información Hospitalaria alas HIS[22]

Esta solución permite la actualización automática de las aplicaciones del Sistema de Información Hospitalaria alas HIS y su servidor de base de datos. Está compuesta por un servidor de actualizaciones que contiene todas las actualizaciones del HIS y una aplicación actualizadora que es la encargada de realizar el proceso y solicitar al servidor nuevas actualizaciones disponibles. La aplicación actualizadora tiene un fichero de configuración que contiene los datos referentes a la fecha y la hora en que se realizará la actualización, puerto y contraseña para conectarse al HIS, nombre del sistema y dirección del servidor de aplicaciones. Esta aplicación hace una copia que contiene todos los datos pertenecientes al sistema que se encontraba en funcionamiento antes de comenzar el proceso de actualización para que en situaciones de fallo en el proceso retome la versión anterior del producto. La transferencia del compactado de actualización se hace mediante SFTP[23] para garantizar la seguridad en la descarga (Reinier Quevedo Batista, 2011).

El mecanismo de actualización definido para este sistema no se adapta al requerido para el SIGESC, pues no cumple con los requisitos analizados y solo automatiza las actividades del sistema para el cual fue diseñado, permitiendo la actualización de aplicaciones sobre plataformas libres.

1.3.4 Mecanismo de actualización deKaspersky Administration Kit8.0

Desarrollada por Kaspersky Lab, Kaspersky Administration Kit es una herramienta que facilita el trabajo del administrador pues permite organizar y monitorizar de forma centralizada la protección de toda una red corporativa, consolidando diferentes capas de protección en un sistema integrado. Está diseñado para controlar y mantener de forma remota y centralizada los sistemas de protección antivirus de Kaspersky Lab (Kaspersky Lab, 2009).

Kaspersky Administration Kit permite entre otras actividades administrar de forma centralizada la descarga de actualizaciones para bases de datos y módulos de aplicaciones y distribuirlas a todos los equipos de red. Este servicio contiene un agente de actualización que consiste en un equipo especial dentro de la red del servidor de administración. Su destino es guardar y distribuir las bases actualizadas, los paquetes de instalación, las tareas y directivas de grupo.

El servicio puede usar la multidifusión IP (Internet Protocol) para distribuir los paquetes de instalación, así como para enviar un paquete de instalación a todos los equipos del grupo que aún no han sido asignados para el establecimiento del producto (Kaspersky Lab, 2011).

[22] HIS: Hospital Information System

[23] SFTP: Secure File Transfer Protocol, Protocolo seguro de trasmisión de archivos

Kaspersky también utiliza los beneficios de los protocolos FTP y HTTP para la descarga de las actualizaciones automáticas. En nuestro país muchas instituciones trabajan con un antivirus muy similar en funcionamiento a Kaspersky denominado SAV, producto desarrollado por cubanos, basado en una estructura cliente-servidor, donde los programas clientes son del tipo Segurmática Antivirus y el servidor es el Servidor Corporativo al cual se conectan éstos, permitiéndoles descargar las actualizaciones y tenerlas disponibles, aunque no cuenta con una herramienta de administración como la de Kaspersky.

El mecanismo de instalación remota y de actualización que utiliza Kaspersky Administration Kit sirve de guía para la construcción del mecanismo propuesto, pero no puede ser utilizada como solución pues es un sistema centrado en un único producto y no cumple con los requerimientos que exige el mecanismo de actualización para el SIGESC.

1.3.5 Mecanismo de actualización para los servidores de Twitter y Facebook

Los servicios web a gran escala como es el caso de Twitter y Facebook cuentan con miles de servidores para gestionar el flujo de cambios enviados por sus millones de usuarios. Para ambas compañías resulta difícil mantener todos estos servidores actualizados con los últimos datos pues puede llevar mucho tiempo y recursos. Ambas redes sociales han implementado nuevas tecnologías para poder ofrecer rapidez y proporcionar a los usuarios la confianza que ofrece un buen servicio en Internet.

Twitter

Los desarrolladores de Twitter implementaban un sistema de servidores GIT[24] con interfaz gráfica, para descargar las últimas versiones del código desde el servidor principal y servirlo. GIT funcionaba efectivamente para Twitter pero a partir de la utilización de cientos de servidores, la escalabilidad comenzó a suponer un problema ya que la actualización de datos se ralentizaba.

La compañía lanzó un sistema denominado Murder basado en el protocolo BitTorrent, es una combinación de *scripts* escritos en Python y Ruby bajo licencia libre Apache integrado con Capistrano[25], que permiten la implementación de un gran número de binarios a través de los centros de datos y convierte a todos los usuarios en extremos o pares que ayudan en la distribución de los mensajes para que todo el proceso sea más eficiente y no se base solo en los servidores, permitiendo así descentralizar el tráfico que genera Twitter. Este sistema logra que Twitter pueda distribuir los archivos de actualización entre sus servidores

[24] GIT es un software de control de versiones diseñado pensando en la eficiencia y confiabilidad del mantenimiento de aplicaciones con un importante número de archivos de código.

[25] Capistrano es una herramienta de código abierto para la ejecución de scripts en varios servidores, y su uso principal es el despliegue de aplicaciones web. Automatiza el proceso de hacer una nueva versión de una aplicación disponible en uno o más servidores web, incluyendo el apoyo a tareas tales como cambiar las bases de datos.

de forma más rápida y eficiente ganando tiempo de respuesta y ahorrando recursos, además elimina el problema que tendría de escalabilidad en el futuro. La implementación de este protocolo resultó que un proceso de despliegue de cuarenta minutos dure apenas doce segundos (Frank, 2010).

Facebook

Diariamente Facebook actualiza cientos de líneas de códigos y el despliegue de estas resultaba un problema preocupante para todos sus desarrolladores. Facebook al igual que Twitter adoptó en su implementación la tecnología BitTorrent la cual le permite movilizar gran cantidad de megabytes (MB) de información de manera rápida, medianamente simple y sobre todo de manera efectiva. El protocolo convierte a cada servidor de Facebook en un compañero que ayuda a distribuir el nuevo código que obtiene actualizado tan pronto como sea posible.

Con el actual sistema, el despliegue entre los servidores de Facebook es más rápido, distribuyendo gran cantidad de MB a miles de equipos en tan solo cuestión de minutos, además el proceso de actualización es igual de rápido sin tener que intervenir personal para llevar a cabo los procedimientos correspondientes (Keating, 2010).

Los mecanismos de actualización concebidos para estas organizaciones no están debidamente publicados, sin embargo el uso de la tecnología BitTorrent para agilizar la transferencia de archivos significa un aporte novedoso en el proceso de actualización. Murder a pesar de ser un sistema con licencia libre y poseer varias características favorables a tener presente en la realización de un mecanismo de actualización, no cuenta con la instalación automática de paquetes de Windows Installer por lo que su diseño no se adapta a los requisitos que necesitan mantenerse con el mecanismo de actualización del SIGESC.

1.4 Características de los sistemas y mecanismos de actualización automática estudiados

En el estudio realizado, se identificó la existencia de tres tendencias principales en la concepción de mecanismos de actualización de aplicaciones (Ver **Figura 1.3**). La primera se evidencia con la solución de Microsoft, y está orientada a la publicación de actualizaciones en el servidor, dejando la responsabilidad a los clientes de realizar el proceso de actualización, mientras que en la segunda tendencia, el servidor tiene conocimiento de sus clientes y es capaz de ejecutar operaciones de actualización en estos, aunque también existe la posibilidad de que el cliente decida cuándo actualizarse, de forma manual o automática, esta variante es aplicada por la solución de Kaspersky y la tercera tendencia basada en una arquitectura de pares implementada por Twitter y Facebook, donde cada nodo de la red usa su capacidad para enviar los archivos de actualización a otros, permitiendo una mejor descentralización y escalabilidad de los servidores.

Partiendo del análisis de los anteriores mecanismos de actualización, se pueden identificar algunas características y conceptos comunes que servirán de utilidad a la construcción de la solución propuesta. Entre los principales aspectos se describen:

- Se identifican los pasos: detección, descarga y activación o aplicación, en el proceso de actualización.
- Descripción de los procesos de actualización a través de documentos en formato XML.
- Publicación de los documentos de actualización de forma centralizada.
- Empleo del modelo cliente-servidor a través de protocolos de comunicación ampliamente utilizados como HTTP, FTP y BITS.
- Empleo de la arquitectura de pares a través del protocolo de comunicación BitTorrent.

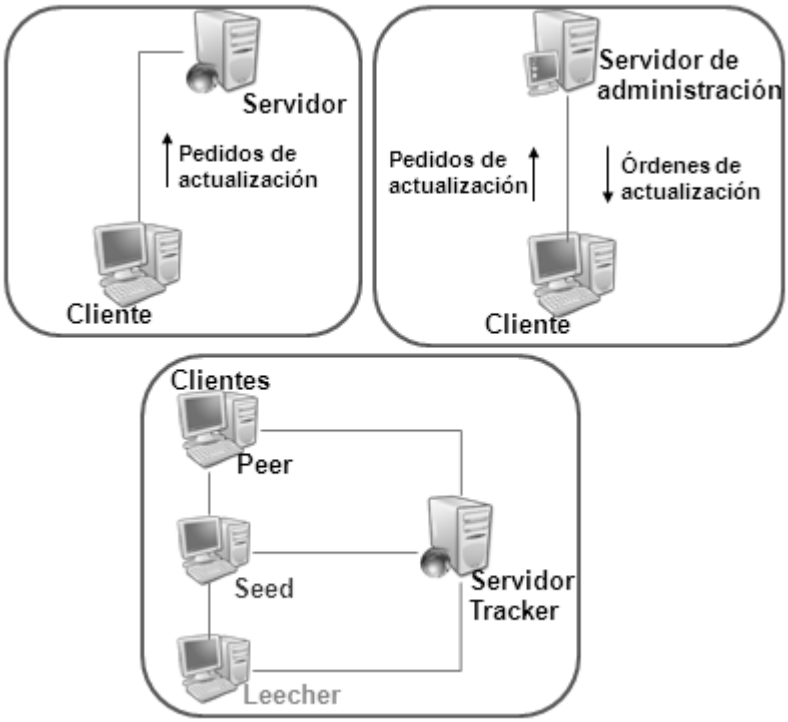

Figura 1.3 - Tendencias en la concepción de mecanismos de actualización de aplicaciones.

Después de analizar las herramientas más adaptables y modelos empleados a nivel internacional y en nuestra universidad que se dedican a la automatización de los procesos de actualización, se ha comprobado que los mismos no cumplen en su totalidad con los requerimientos que propone el SIGESC para constituir un mecanismo de actualización acorde a sus necesidades actuales, sino que estas soluciones proveen una acumulación de experiencias útiles para el desarrollo de una solución en la presente investigación.

Siguiendo con la línea de desarrollo del SIGESC se proponen la metodología de desarrollo, el lenguaje de modelado, el lenguaje de programación y las herramientas que serán utilizadas en la constitución de una solución, como parte del mecanismo de actualización, que proveerá al SIGESC de una mejoría en el proceso de mantenimiento y soporte.

1.5 Metodología de Desarrollo de Software.Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo (RUP[26]) es una metodología tradicional de software que no establece pasos estrictamente obligatorios sino un conjunto de guías que se adaptan a las necesidades de los desarrolladores. RUP está fundamentado en seis primicias claves:

- Adaptar el proceso a las necesidades del cliente para establecer una buena comunicación con él.
- Demostrar valor iterativamente pues el proyecto debe realizarse por iteraciones, donde cada iteración es examinada por los inversores para medir la calidad y estabilidad del producto.
- La colaboración entre equipos es primordial ya que se hace necesario que fluya la comunicación en todos los sentidos para que el trabajo no se vea afectado en un equipo donde trabajan de forma conjunta múltiples personas para lograr el desarrollo del software.
- Un alto nivel de abstracción permite discusiones sobre diversos niveles y soluciones arquitectónicas que pueden estar acompañadas por las representaciones visuales de la arquitectura mediante un lenguaje de modelado como UML.
- El control de la calidad no se debe realizar solo al concluir una iteración sino en cada uno de los aspectos de la realización del software, por lo cual, asegurar la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

Integrar RUP con el Lenguaje Unificado de Modelado constituye el procedimiento estándar más utilizado para el análisis, documentación e implementación de sistemas orientados a objetos.

1.6 Lenguaje de Modelado. Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML[27]) es el lenguaje más destacado y utilizado en el modelado de sistemas de software. UML al ser un lenguaje gráfico, permite visualizar, especificar, construir y documentar el sistema que se propone desarrollar. Ofrece un estándar para detallar un plano del sistema (modelo), incluyendo características conceptuales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguaje de programación, esquemas de bases de datos y componentes reutilizables. Se puede emplear en el desarrollo de un software como soporte a una

[26]RUP, Rational Unified Process

[27]UML, Unified Model Language

metodología de desarrollo como RUP, pero no se especifica qué metodología o proceso se debe usar (Larman, 2003).

1.7 Tecnologías y herramientas utilizadas

Posterior al análisis de las tendencias y conceptos importantes en el desarrollo de aplicaciones que permitan actualizaciones automáticas, se procedió a realizar una selección de las tecnologías y herramientas a utilizar para darle cumplimiento al objetivo general de la investigación.

1.7.1 Plataforma Microsoft .NET

La Plataforma .NET provee las herramientas y tecnologías requeridas para el desarrollo rápido de aplicaciones, proporcionando una manera efectiva y la vez cómoda para implementar soluciones. Entre los lenguajes de programación que se han adherido a esta plataforma se encuentra el C#, Visual Basic, C++, J#, Pascal, entre otros conformando un amplio grupo de lenguajes. Para su óptimo aprovechamiento se desarrolló Visual Studio 2003 que ha evolucionado hasta las actuales versiones de Visual Studio 2012.

Se seleccionó esta infraestructura ya que provee un entorno coherente de programación orientada a objetos y un entorno de ejecución de código que reduce los costos de la implementación de software y los conflictos de versiones. Aporta al programador mejoras en cuanto a aspectos como la seguridad, el uso de la firma y los certificados digitales, importante característica para el proceso de distribución de actualizaciones. Igualmente ofrece un conjunto de tecnologías que facilitan el desarrollo de aplicaciones como el lenguaje de programación C#, bibliotecas de código reutilizables, y el uso de tecnologías como Windows Communication Foundation (WCF) (Sharp, 2007).

1.7.2 .Net Framework v4.0

Microsoft .NET Framework 4.0 es una plataforma de software que integra disímiles tecnologías facilitando el trabajo entre diferentes lenguajes de programación y librerías, con el objetivo de concebir aplicaciones e integrarlas a otros sistemas previamente creados. Proporciona además un modelo de programación independiente del lenguaje para todas las capas o niveles de una aplicación y una interoperabilidad transparente entre tecnologías (Makey, 2010).

Dentro de las tecnologías que forman parte del Framework 4.0 se destaca Windows Communication Foundation (WCF). WCF es un modelo de programación unificado para crear aplicaciones distribuidas y aplicaciones con Arquitectura Orientada a Servicios (SOA), avalando la interacción y la conectividad entre diferentes plataformas e innumerables dispositivos. Con el uso de este modelo se garantiza la comunicación entre aplicaciones bajo un único estilo de desarrollo, mediante mecanismos que son

descritos a niveles de configuraciones, el uso de contratos y el descubrimiento de los servicios (Sharp, 2007).

1.7.3 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 brindan, en las cuales un aspecto muy importante radica en la verdadera riqueza del lenguaje.

1.7.3.1 El lenguaje C#

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. La plataforma .Net permite utilizar este lenguaje para crear aplicaciones cliente para Windows tradicionales, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, y otras tareas.

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.

Este lenguaje ha alcanzado gran auge por su sencillez y se utiliza para desarrollar gran cantidad de aplicaciones desde mediano a gran tamaño, disminuyendo el esfuerzo de programadores en actividades de rutina. Es gratis tanto el C# como la plataforma que lo soporta, siendo un lenguaje debidamente estandarizado, resultando por tanto adecuado para las aspiraciones y necesidades de este trabajo.

1.7.4 Herramientas a utilizar

1.7.4.1 Herramienta CASE[28]. Visual Paradigm for UML 8.0 Enterprise Edition

Visual Paradigm for UML (VP-UML) es una herramienta CASE UML diseñada para la ayuda al desarrollo de software y que brinda soporte al modelado visual con UML 2.1. Posee una gran cantidad de módulos para facilitar el trabajo de desarrollo de un software y garantizar la calidad del producto final. VP-UML soporta los principales estándares de la industria tales como SysML[29], BPMN[30], XMI[31], UML, etc. Ofrece

[28] Ingeniería de Software Asistida por Ordenador. Computer Aided Software Engineering.

[29] SysML: Systems Modeling Language, Lenguaje de modelado de sistemas

[30] BPMN: Business Process Modeling Notation, Notación para el Modelado de Procesos de Negocio.

[31] XMI o XML: Metadata Interchange, Intercambio de metadatos

un completo conjunto de herramientas de los equipos de desarrollo de software necesario para los requisitos de la captura, software de planificación, la planificación de controles, el modelado de clases, modelado de datos, etc. Sus principales características son:

- Brinda la posibilidad de crear los artefactos necesarios para la confección de un software cumpliendo además con el estándar UML 2.1.
- Brinda numerosos estereotipos a utilizar lo cual permite que los diagramas se entiendan mejor.
- Puede generar código a partir de diagramas para plataformas como .Net, Oracle, así como obtener diagramas a partir de código.
- Se integra fácilmente con ambientes de desarrollo integrados como Visual Studio.
- Brinda la posibilidad de documentar todo el trabajo sin tener que utilizar herramientas externas.
- Disponible en múltiples sistemas operativos como Microsoft Windows XP y Linux.

1.7.4.2 Microsoft Visual Studio 2010

Microsoft Visual Studio 2010 es una herramienta de desarrollo que engloba un conjunto de funcionalidades que facilitan y mejoran la construcción de aplicaciones para entornos de escritorio, web o para servicios Windows.

La integración de funciones útiles de versiones anteriores, junto con novedosas características incluidas en este producto, proporcionan al entorno una madurez nunca antes adquirida dando como resultado una plataforma altamente integrada y preparada para el desarrollo de aplicaciones escalables y reutilizables.

Microsoft Visual Studio 2010 posee una serie de características nuevas que dinamizan no solo el trabajo en el entorno sino también el aprendizaje y entrenamiento de los desarrolladores. Algunas de estas características son (Makey, 2010):

- *Multitargeting Support*: Brinda la posibilidad de compilar distintas versiones del Framework (2.0, 3.0, 3.5, 4.0)
- *Intellisense*: Facilita la programación con clases y métodos permitiendo el completamiento de código sin tener que recordar exactamente los tipos de datos, propiedades o estructuras buscadas. Simplifica la búsqueda de tipos de datos ya que soporta las búsquedas basadas en mayúsculas.

1.8 Conclusiones

Como resultado del estudio realizado en este capítulo, se puede concluir que los sistemas existentes que gestionan procesos asociados a las actualizaciones automáticas para aplicaciones informáticas no cumplen con los requisitos necesarios que demandan los procesos de mantenimiento y soporte del SIGESC. A pesar de ser sistemas que poseen funcionalidades determinantes para que un proceso de actualización trabaje de manera correcta, no proveen un ambiente de configuración que permita

adaptarlos al entorno de despliegue de las aplicaciones a actualizar. Por lo que se evidenció la necesidad de desarrollar un mecanismo que se ajuste más a las necesidades propias del SIGESC facilitando en gran medida el mantenimiento y soporte del mismo.

Posterior al estudio del arte se definió para la realización del sistema propuesto la metodología de desarrollo RUP, la herramienta de modelado Visual Paradigm para la confección y diseño del sistema de actualización mediante el lenguaje UML, el lenguaje de programación C# y las tecnologías que brinda la plataforma .NET 4.0 para la implementación del mismo mediante el entorno de desarrollo integrado Visual Studio 2010.

Capítulo II: Construcción de la solución

2.1 Introducción

El presente capítulo tiene como objetivo la descripción de las actividades que serán objeto de automatización por parte del mecanismo de actualización automática para el SIGESC, prestando gran atención a la situación problemática que dio origen a desarrollar el sistema que se propone. Debido a la poca claridad de los procesos del negocio se realiza un Modelo de Dominio y se hace un análisis de los requerimientos funcionales y no funcionales para lograr un mejor entendimiento del sistema. Se especifican los casos de uso relevantes del sistema, se describen las clases del diseño y se elabora el modelo de implementación del sistema.

2.2 Objetos de automatización

Los problemas que implica el desempeño manual de las actividades de actualización de las aplicaciones que componen el SIGESC y la importancia que tiene el mantenimiento y soporte de estos, crean la necesidad de automatizar dichas actividades a través de un mecanismo de actualización.

El mecanismo de actualizaciones automáticas para el SIGESC incluye una mejora de los requisitos del mecanismo de actualización existente analizados en el Capítulo 1, que permitirá la distribución e instalación de manera automática de paquetes MSP y solo la distribución de paquetes MSI. El mecanismo permitirá la transferencia de estos archivos a través del protocolo BitTorrent, escogido por los beneficios que brinda en cuanto a rapidez, escalabilidad, eficiencia y confiabilidad en la descarga. Para implementar este protocolo se utilizarán las funcionalidades que ofrece la librería de clases que proponen los desarrolladores de BitTorrent denominada MonoTorrent.

Los problemas que introduce BitTorrent son minimizados a través de la implementación de las siguientes medidas de contingencia:

- Se manipularán mecanismos a niveles de transporte que aprueban el cifrado y la firma digital de los datos que viajan por el canal de comunicación por medio de la creación de certificados, evitando que la información no sea alterada ya que archivos no deseados como virus, gusanos y troyanos pueden ingresar a los ordenadores como ficheros Torrent.
- Se utilizará la tecnología WCF de .Net que permitirá además la transferencia de los archivos Torrent por canales TCP ya que BitTorrent no es un protocolo totalmente distribuido pues los ficheros Torrent tienen que ser transmitidos a través de canales externos. WCF es una nueva tecnología para la creación de sistemas interconectados que se basa en la arquitectura orientada a servicios. WCF

unifica una gran variedad de funcionalidades de aplicaciones distribuidas en una arquitectura organizada y extensible, que abarca transporte, sistemas de seguridad, patrones de mensajería, sistema de codificación, topología de red y modelos de alojamiento (Fernández, 2010).

- El problema de la gestión del ancho de banda de BitTorrent significa una dificultad para el mecanismo pues en las estaciones del SIGESC se hace un uso extensivo de la red del centro, sin embargo las librerías de clases que proponen los desarrolladores de BitTorrent para utilizar las funcionalidades que ofrece el protocolo uTP están implementadas solo para los lenguajes Python y C++, resultando incompatibles con el lenguaje seleccionado para el desarrollo de la solución.

Los problemas que ocasiona la falta de disponibilidad de las actualizaciones serán minimizados con la implementación de "Servicios de Windows" los que funcionarán adecuadamente en cualquier momento, con el objetivo de que las actualizaciones sean desplegadas e instaladas en tiempo. Estos "Servicios de Windows" son aplicaciones de larga ejecución que corren en un segundo plano, son iniciados automáticamente por el sistema operativo y por lo general cuentan con un alto grado de autonomía. Este tipo de aplicaciones no posee interfaz gráfica para interactuar con el usuario por lo que se dificulta el control de todas las operaciones que las mismas realizan y con esto se maximiza la posibilidad de que se encuentren en un estado incoherente sin que los administradores del sistema tengan conocimiento, por lo que se realizará el control y supervisión de los mismos a través de una aplicación de administración para cada servicio que se conectará al mismo utilizando la librería que propone .Net para la administración de este tipo de aplicaciones, System.ServiceProcess.dll, a través de las funcionalidades que ofrece la clase ServiceController.

El estado coherente entre las aplicaciones es uno de los objetivos más importantes en la solución que se propone desarrollar pero uno de los más complejos, los factores que afectan la coherencia de las aplicaciones que componen el SIGESC serán minimizados a través del control automático de las actualizaciones que se implantan en cada estación de un centro de atención de emergencias con el propósito de detectar que no existan diferentes versiones de una misma aplicación ejecutándose.

2.3 Descripción del mecanismo de actualización

La automatización del mecanismo de actualización del SIGESC fusiona dos tendencias en la concepción de mecanismos de actualización de aplicaciones estudiadas en el capítulo anterior, donde la primera está orientada a la publicación de actualizaciones en el servidor, dejando la responsabilidad a los clientes de realizar el proceso de actualización y en la segunda cada nodo de la red usa su capacidad para enviar los archivos de actualización a otros, permitiendo una mejor descentralización y escalabilidad de los servidores y agrega nuevas funcionalidades al proceso de actualización en comparación con la primera tendencia.

Se han dividido las funcionalidades de este mecanismo en dos partes importantes con responsabilidades definidas de la siguiente forma:

- La funcionalidad “Proveedor de Actualización” es el servicio que debe estar ubicado en uno o varios servidores cuya función principal es publicar los paquetes de actualización, interpretar la información que proveen los mismos, almacenarla en una base de datos y atender los pedidos de actualización por parte de los clientes u otros servidores. Mediante el uso del protocolo BitTorrent se debe permitir la creación de archivos Torrent a partir de estos paquetes y compartirlos de forma más rápida. Dadas las necesidades del SIGESC y con el objetivo de proveer una distribución efectiva de las actualizaciones hacia todos los centros de atención de emergencias ubicados en Venezuela, el “Proveedor de Actualización” puede descargar las actualizaciones desde otro servidor permitiendo propagarlas hacia otro nivel de la red de distribución de actualizaciones del sistema. Además permite el control global de todos los clientes o servidores en la red de distribución de actualizaciones, obteniendo información del estado de actualización por cada tipo de aplicación que se instale en el cliente. Las actualizaciones serán publicadas en una estructura de carpetas definida de la siguiente forma:
    - Para los paquetes de actualización: *Disco Local (Ej. C):Archivos Comunes\ActualizacionSIGESC\Version_#*
    - Para los Torrent de cada paquete de actualización: *Disco Local (Ej. C): Archivos Comunes \ActualizacionSIGESC\ Version_#\Torrents*
- La funcionalidad “Actualizador de Aplicaciones”, localizada en los clientes o estaciones de trabajo de un centro de atención de emergencias, será la encargada de iniciar el proceso de comunicación con el “Proveedor de Actualización”, debe permitir la descarga de los archivos Torrent de las aplicaciones que necesita actualizar en la estación y acceder a los archivos originales mediante el protocolo BitTorrent, que le permite compartir partes de la información que se está descargando con otros clientes y saber cuáles poseen piezas de la información que descarga aparte del servidor (semilla inicial) que se las provee. Esta funcionalidad debe permitir además instalar las actualizaciones, hacer verificaciones en el sistema y en el “Proveedor de Actualización”con el objetivo de que el mecanismo se realice de forma rápida y confiable. Los archivos Torrent y los paquetes de actualización son descargados hacia carpetas especiales en el sistema operativo con el objetivo de que los archivos sean utilizados por el servicio de actualización y después de la instalación sean eliminados automáticamente.

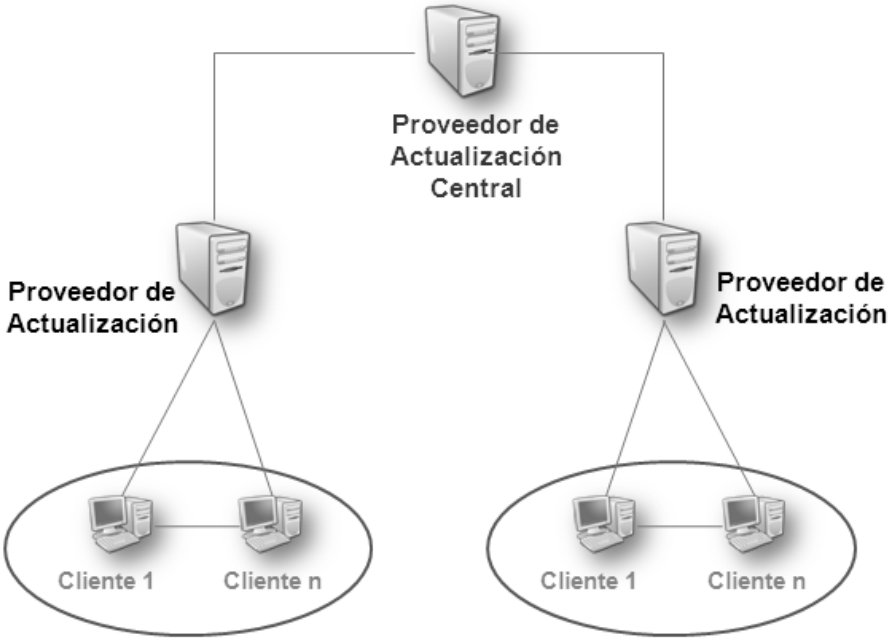

Figura 2.1–Escenario de distribución del sistema de actualización.

2.4 Descripción del sistema de actualización

Para el desarrollo del sistema de actualización que forma parte del mecanismo presentado se propone un diseño híbrido[32], debido a la utilización del protocolo BitTorrent, donde se combinan lo mejor de la arquitectura de pares y el modelo cliente-servidor (Ver **Figura 2.2**). Esta alianza debe permitir que las desventajas del primero se solapen con las ventajas del segundo y viceversa, trabajando conjuntamente para generar eficiencia y fortaleza a las comunicaciones.

Figura 2.2 Modelo Híbrido.

[32]Híbrido: Es la unión de dos entes, en este caso es la composición de dos modelos, P2P y Cliente/Servidor.

El sistema debe permitir que la información sea publicada y administrada en uno o varios servidores que constituyen Tracker o rastreadores de una red BitTorrent, los cuales crean la primera semilla de contenido, fusionando dichas actividades a través de un "Servicio de Windows". Los clientes y los proveedores de actualización de un nivel intermedio, observados en la figura 2.1, pueden funcionar como los pares, semillas o sanguijuelas en dependencia del estado de la descarga, de modo que interactúen con un servidor rastreador haciendo pedidos de recursos a través de la lógica cliente/servidor y a la vez comparten el contenido de la descarga con otros nodos que hacen uso de la red BitTorrent. El cliente es la aplicación que debe permitir actualizar de manera automática las aplicaciones del SIGESC, funcionando estas actividades mediante un "Servicio de Windows" con el objetivo de lograr la disponibilidad del mecanismo. Los servicios serán gestionados a través de una aplicación de administración la cual presentará la información haciendo uso de la biblioteca de clases de .NET System.ServiceProcess.dll y se controlará cuáles han sido las versiones de las aplicaciones que ha descargado cada estación de trabajo desde el servidor que se encarga de proveer las actualizaciones.

2.5 Modelo de dominio

El modelo de dominio es una representación visual de los conceptos y objetos del mundo real, significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema. Se modelan conceptos del mundo real, no de los componentes de software. Una clase conceptual puede ser una idea o un objeto físico (Larman, 2003).

El modelo de dominio se representa en UML con un diagrama de clases en el que se muestran:

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

El modelo de dominio se realiza si no se logran determinar los procesos de negocio con fronteras bien establecidas donde se logren ver claramente, quiénes son las personas que realizan cada actividad del negocio, quiénes son los beneficiados con cada uno de estas tareas, pero además quiénes son las personas que desarrollan las actividades en cada uno de estos procesos. El sistema de actualización para el SIGESC forma parte de un mecanismo 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 en la automatización del proceso de actualización de aplicaciones.

2.5.1 Conceptos fundamentales del dominio

Paquete de Actualización: Contiene todos los componentes necesarios (dígase .MSI o .MSP) para dar comienzo al flujo de actividades que engloba una actualización.

Repositorio: Contiene el directorio de archivos, el cual contendrá las versiones pasadas y nuevas de los productos que se encuentran en carpetas creadas para guardar las mismas, en este directorio se realizarán las salvas continuas de las versiones.

Servidor de Actualizaciones: Representa el o los servidores de actualizaciones, implementa el servicio que responderá las solicitudes de los clientes con información detallada sobre actualizaciones nuevas y permitirá la publicación y descarga de los paquetes de actualización por los clientes u otros servidores.

Archivo Torrent: Contiene metadatos que indican todos los paquetes de actualización que pueden descargarse a través de él, como nombres, tamaños, *hashes* de todas sus partes y la dirección del servidor Tracker de BitTorrent o el servidor de actualizaciones donde fueron creados.

Servicio Actualizador Cliente: El servicio actualizador cliente es el que se encontrará en las máquinas donde estén las aplicaciones que son objeto de actualización, permitirá verificar existencia de nuevas actualizaciones de los productos, descargará el archivo Torrent y a través de él descargará los paquetes de actualización, los que luego instalará en la estación.

Base de datos de Actualización: Representa un archivo ubicado en el servidor de actualizaciones, es donde se almacena la información que ha sido interpretada en el paquete de actualización para facilitar el desarrollo de las funcionalidades de la solución.

Conexión: Encargada de describir de qué manera se comunican el servidor de actualizaciones y el servicio actualizador cliente. La conexión determina el método de transporte, la codificación del mensaje, los requerimientos de seguridad, las sesiones confiables y las transacciones.

Aplicaciones: Sistema informático que va a ser actualizado por el servicio actualizador cliente.

2.5.2 Diagrama del modelo de dominio

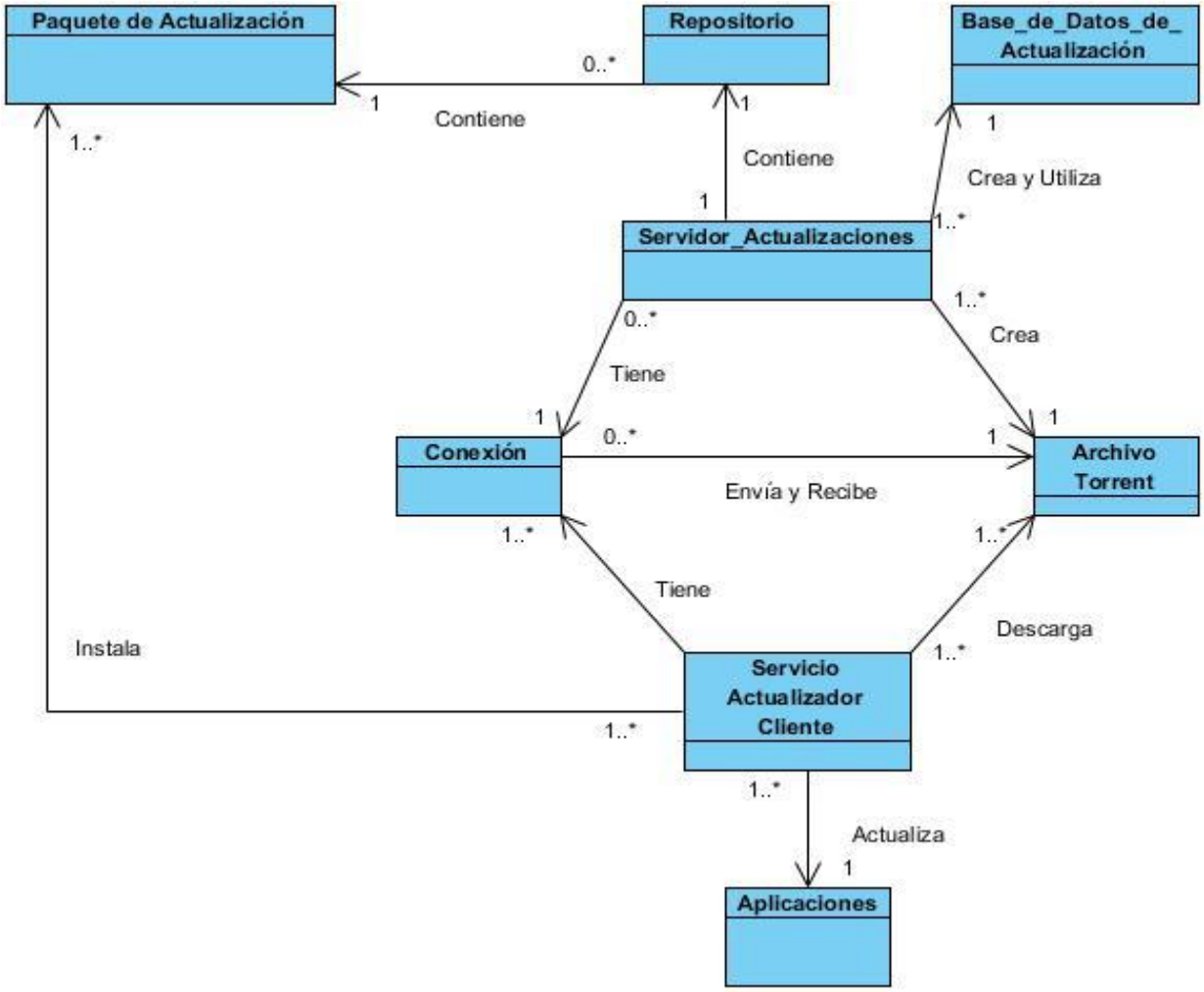

Figura 2.3 - Modelo del Dominio

2.6 Requisitos Funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir y que definen el comportamiento interno del software. Estos describen los servicios que se espera que el sistema cumpla para satisfacer las necesidades del usuario (Pressman, 2005).

Los requisitos funcionales que el sistema debe cumplir son los siguientes:

RF 1 Iniciar servicios.

RF 2 Administrar servicios.

RF 3 Configurar aplicación servidor.

RF 4 Publicar actualización.

RF 5 Guardar datos de la actualización en la base de datos.

RF 6 Crear archivos Torrent por cada paquete de actualización.

RF 7 Activar la función de rastreador de BitTorrent.

RF 8 Activar la función de semilla de BitTorrent.

RF 9 Proveer los paquetes de actualización a los clientes.

RF 10 Controlar estado de actualización de las aplicaciones.

RF 11 Configurar aplicación cliente.

RF 12 Comprobar qué aplicaciones del SIGESC están instaladas y la versión de cada una en la estación.

RF 13 Informar si no existen aplicaciones del SIGESC instaladas en la estación.

RF 14 Comprobar si en el servidor de actualizaciones hay nuevas actualizaciones.

RF 15 Descargar archivos Torrent de los paquetes de actualización.

RF 16 Descargar paquetes de actualización a partir de archivos Torrent.

RF 17 Verificar si el producto al que se le va a hacer la actualización está en ejecución.

RF 18 Iniciar actualización de aplicaciones.

RF 19 Eliminar archivos temporales de actualización.

2.7 Requisitos No Funcionales

Los requisitos no funcionales son propiedades o cualidades que el sistema debe poseer. Forman una parte significativa de la especificación y son importantes para que clientes y usuarios, puedan valorar las características no funcionales de la aplicación desarrollada. Si se conoce que un sistema cumple con las funcionalidades requeridas, las propiedades no funcionales (como cuán usable, seguro, conveniente y agradable es este) pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.

RNF 1 Software

- Framework 4.0 de .NET con los sistemas operativos Windows XP Service Pack 3 para las estaciones de trabajo y para los servidores Windows Server a partir de la versión del 2003.
- El Cortafuegos de las estaciones y del servidor debe estar activado y configurado para que permita las conexiones de los nodos clientes con el nodo servidor o las conexiones entre servidores.

RNF 2 Hardware

- Se debe disponer de al menos un servidor para desplegar el servicio. Se sugiere para una disposición óptima hacer uso de dos o más servidores para el respaldo de los servicios.

- La capacidad de memoria RAM[33] para el servidor debe ser de 1 Gigabytes (GB) en adelante y los ordenadores clientes deberán poseer como mínimo 512 MB.
- Se deberá contar con procesadores de 2.0 Gigahertz (GHz) y disco duro de al menos 10 GB libres para el servidor y 1 GB libre para el cliente.

RNF 3 Seguridad

- Se manejarán mecanismos a niveles de transporte que aprueban el cifrado y la firma digital de la información que viaja por el canal de comunicación, evitando que sea corrompida.
- El sistema podrá ser utilizado en todo momento.

2.8 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 software deberá realizar, así como la relación que los mismos guardan entre sí. A continuación se definen los actores del sistema y los diagramas de paquetes y casos de uso (CU) del sistema propuesto.

2.8.1 Actores del sistema

Tabla 2.1 Actor que interviene en el Sistema

Nombre	Justificación
Administrador Informático (AI)	Personal encargado de configurar y asistir el flujo de eventos que comprende el mecanismo de actualización.
Sistema Operativo	Representa al encargado de iniciar los servicios que engloban las acciones a realizarse durante la actualización de las aplicaciones.
Aplicación cliente	Representa el encargado de iniciar acciones en el nodo servidor que consisten el envío de mensajes a través de canales de comunicación TCP.

2.8.2 Diagrama de casos de uso del sistema

Los diagramas de casos de uso son representaciones gráficas en las que se muestran los requisitos funcionales que se esperan de un sistema y su relación con el entorno. Los casos de uso son utilizados básicamente en el proceso de modelado de sistemas. Fue necesario realizar un diagrama de paquetes para agrupar los casos de uso por funcionalidades debido a que se distinguen los requisitos funcionales

[33] RAM: Random Access Memory, Memoria de acceso aleatorio.

de un nodo que provee información y los del nodo que la consume y de esta forma conseguir entender mejor el funcionamiento de la solución.

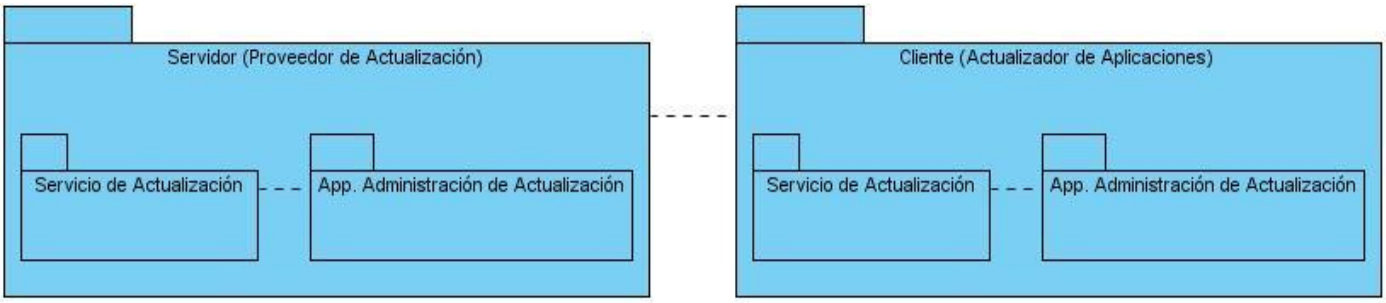

Figura 2.4 - Diagrama de Paquetes.

Los diagramas donde se aprecia la relación existente entre actores y los casos de uso se representan a continuación:

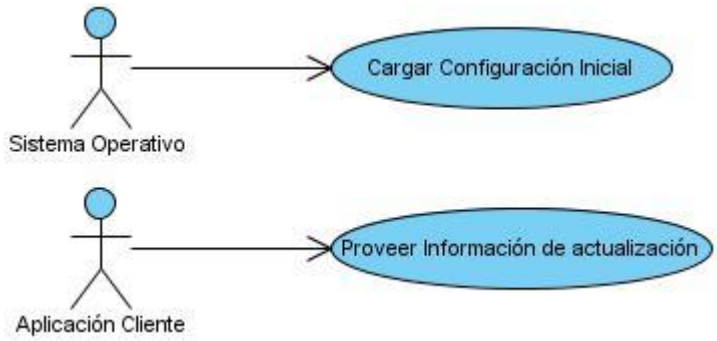

Figura 2.5 -Diagrama de CU del Paquete Servidor- Paquete Servicio de Actualización.

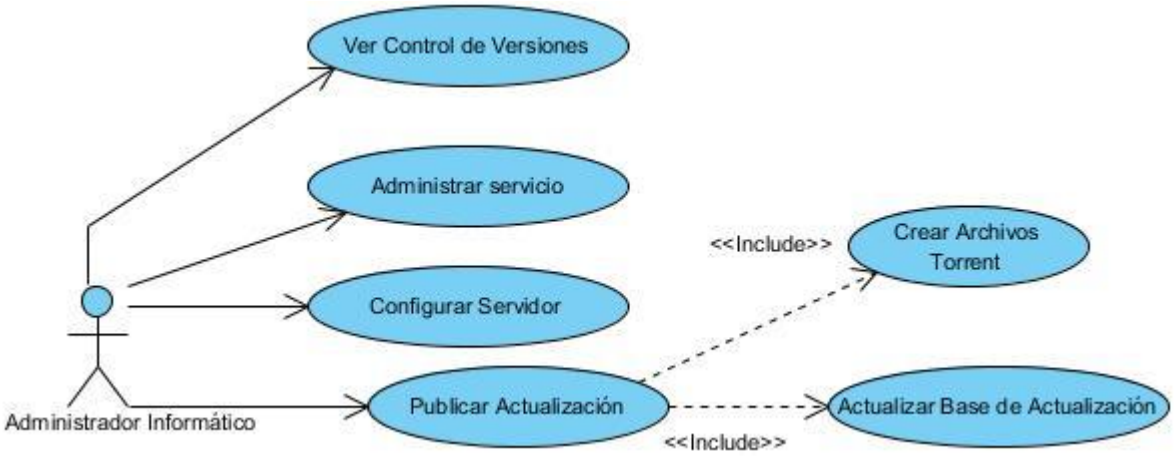

Figura 2.6 Diagrama de CU del Paquete Servidor- Paquete App. Administración de Actualización.

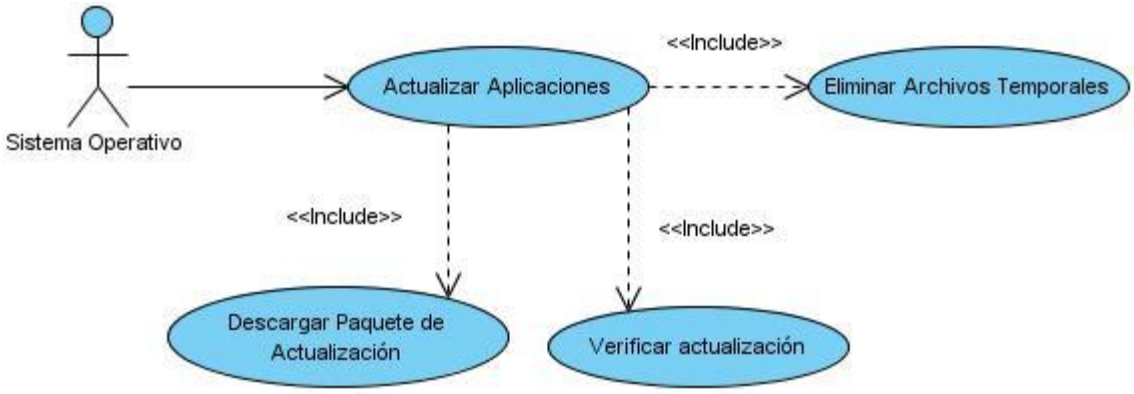

Figura 2.7 Diagrama de CU del Paquete Cliente - Paquete Servicio de Actualización.

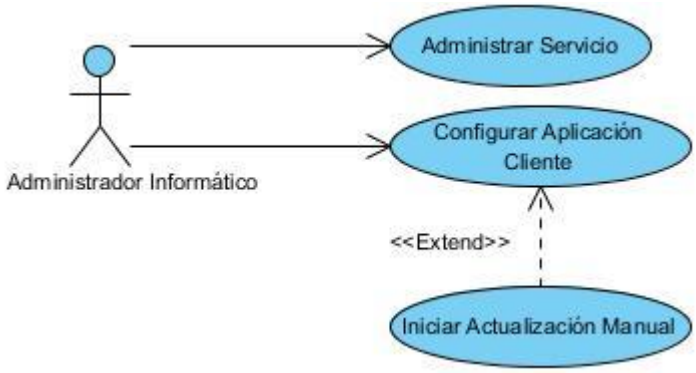

Figura 2.8 Diagrama de CU del Paquete Cliente - Paquete App. Administración de Actualización.

2.8.3 Especificación de casos de uso

El objetivo principal de describir los casos de uso de un sistema es referir las funcionalidades asociadas a cada uno, detallando el flujo de eventos. El diagrama de casos de uso del sistema brinda una representación gráfica de elementos importantes que los programadores necesitan comprender rápidamente pero sin una descripción de los casos de uso sería insuficiente. Las especificaciones extendidas de los casos de uso se encuentran en los anexos.

Descripción del Caso de Uso: Cargar Configuración Inicial

Caso de Uso:	Cargar Configuración Inicial
Propósito:	Iniciar el servicio de actualización en el nodo servidor con los datos iniciales que necesita para su correcto funcionamiento.
Resumen:	Se inicia el servicio de actualización en el nodo servidor cargando el tipo de servidor a desplegar e inicia el rastreador y la semilla de BitTorrent.
Referencia	**RF 1**

Descripción del Caso de Uso: Proveer Información de actualización.

Caso de Uso:	Proveer Información de actualización
Propósito:	Facilitar a los clientes las actualizaciones solicitadas.
Resumen:	El CU inicia cuando el sistema recibe peticiones de la aplicación cliente con el fin de encontrar nuevas actualizaciones disponibles para las aplicaciones y procesa todos los comandos necesarios para proveer la información al cliente.
Referencia	**RF 9**

Descripción del Caso de Uso: Administrar servicio.

Caso de Uso:	Administrar servicio
Propósito:	Administrar el servicio, permitiendo actualizar su estado, detenerlo o iniciarlo.
Resumen:	Obtiene los datos del servicio, a partir del estado en que este se encuentre es posible detener o iniciar el servicio.
Referencia	**RF 2**

Descripción del Caso de Uso: Configurar Servidor

Caso de Uso:	Configurar Servidor
Propósito:	Proveer una configuración nueva para el servicio de actualización.
Resumen:	El Cu inicia cuando el AI accede a la opción “Configuración”. Este modifica las variables necesarias para establecer la forma en la que se va a comportar el sistema utilizando esta configuración.
Referencia	**RF 3**

Descripción del Caso de Uso: Publicar Actualización.

Caso de Uso:	Publicar Actualización
Propósito:	Publicar las actualizaciones en el repositorio del servidor.
Resumen:	El caso de uso inicia cuando el AI accede a la opción “Publicación” para publicar las actualizaciones en el servidor.
Referencia	**RF 4**

Descripción del Caso de Uso: Crear Archivos Torrent.

Caso de Uso:	Crear Archivos Torrent
Propósito:	Crear los archivos Torrent por cada paquete de actualización publicado en el servidor.

Descripción:	El caso de uso inicia una vez que se haya configurado el servidor y se publiquen informaciones nuevas en el repositorio, el sistema lleva a cabo la creación de los archivos Torrent por cada paquete de extensión MSI o MSP y el caso de uso termina.
Referencia	**RF 7**

Descripción del Caso de Uso: Actualizar Base de Actualización.

Caso de Uso:	Actualizar Base de Actualización
Propósito:	Guardar los datos de las actualizaciones liberadas reflejados en los paquetes MSP.
Descripción:	El caso de uso inicia cuando el AI ha publicado los paquetes de actualización y el sistema interpreta la información de los mismos guardando los datos de la actualización en la base de datos del servidor.
Referencia	**RF 5**

Descripción del Caso de Uso: Configurar Aplicación Cliente.

Caso de Uso:	Configurar Aplicación Cliente.
Propósito:	Proveer una configuración nueva para el servicio de actualización en el cliente.
Descripción:	El caso de uso inicia cuando el AI introduce el comando para configurar la aplicación. Este modifica o crea una nueva configuración, el sistema guarda la configuración y el caso de uso termina.
Referencia	**RF 11**

Descripción del Caso de Uso: Actualizar Aplicaciones.

Caso de Uso:	Actualizar Aplicaciones.
Propósito:	Permitir la actualización de las aplicaciones del SIGESC instaladas en una estación.
Descripción:	El caso de uso inicia cuando el sistema lleva a cabo la actualización de las aplicaciones programada cada cierto intervalo de tiempo y el caso de uso termina.
Referencia	**RF 12, RF 13, RF 14, RF 17 - RF 19**

Descripción del Caso de Uso: Descargar Paquete de Actualización.

Caso de Uso:	Descargar Paquete de Actualización.
Propósito:	Descargar los paquetes de actualización de las aplicaciones.
Resumen:	El caso de uso inicia cuando el sistema ha comprobado la existencia de nuevas actualizaciones disponibles en la dirección de destino configurado y descarga los archivos Torrent y luego los ejecuta para a través de estos descargar los paquetes publicados.
Referencia	**RF 15, RF 16**

Descripción del Caso de Uso: Ver Control de Versiones.

Caso de Uso:	Ver Control de Versiones.
Propósito:	Facilitar a los usuarios la información acerca del estado de las actualizaciones realizadas a las aplicaciones del SIGESC en un centro.
Resumen:	El caso de uso inicia cuando el usuario accede a la opción Control de Versiones y se muestran los datos de todas las aplicaciones del SIGESC instaladas en las máquinas de un centro.
Referencia	**RF 10**

Descripción del Caso de Uso: Eliminar Archivos Temporales.

Caso de Uso:	Eliminar Archivos Temporales
Propósito:	Eliminar los archivos de actualización que se crean en las carpetas especiales en el proceso de actualización en las estaciones de trabajo.
Resumen:	El caso de uso inicia cuando el sistema elimina los archivos de actualización que se han copiado a la estación de trabajo durante el proceso de actualización y ya no serán utilizados por el sistema.
Referencia	**RF 21**

2.9 Arquitectura del sistema

"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución" (IEEE[34], 2000).

[34]IEEE: Institute of Electrical and Electronics Engineers, Instituto de Ingenieros Eléctricos y Electrónicos.

Existe una amplia gama de modelos arquitectónicos probados y reconocidos por las ventajas de su aplicabilidad y funcionamiento, pero en ocasiones un solo estilo no da soporte al producto a construir. En algunos casos los desarrolladores y arquitectos se ven en la necesidad de emplear más de uno para dar solución a las peticiones y necesidades del software a desarrollar.

En el epígrafe 2.4 se propuso una arquitectura híbrida para la construcción del sistema de actualización, la cual está compuesta por los elementos más positivos de la arquitectura Cliente/Servidor y la P2P. El objetivo fundamental es solucionar una de las problemáticas de la investigación en cuanto al despliegue de las actualizaciones, debido a que el equipo de desarrollo se encuentra en la UCI necesitando proveer de forma automatizada las actualizaciones hacia los centros de atención a emergencias ubicados en Venezuela, mediante una aplicación que ocupe el rol de proveedor y otra que se desempeñe como consumidor de esa información, logrando así separar la lógica de funcionamiento del cliente de la del servidor, evitando centralizar la mayor parte de la solución.

Partiendo de esta premisa, se puede afirmar que la arquitectura conformada para el sistema está compuesta por un conjunto de estilos, utilizando para lograr la comunicación entre cliente y servidor el protocolo BitTorrent y la tecnología WCF (Ver **Figura 2.9**).

Figura 2.9 Arquitectura Cliente/Servidor.

2.9.1 Arquitectura de los módulos

Conjuntamente con la elección de la arquitectura del sistema, se determinó el uso del estilo arquitectónico tres capas con el objetivo de estructurar la implementación para soportar requerimientos complejos operacionales y garantizar mantenibilidad, reusabilidad, robustez y seguridad.

La arquitectura tres capas propuesta para los módulos está conformada específicamente por las capas: Presentación, Negocio, Acceso a Datos, donde cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior, donde las capas intermedias resultan transparentes para el resto de las capas. El sistema contará con una aplicación cliente y otra servidor, los módulos definidos para estas aplicaciones son el servicio de actualización, que dadas las características de los servicios de Windows no posee la capa de Presentación y el módulo aplicación de administración del servicio de actualización (Ver **Figura 2.9** y **Figura 2.10**).

Figura 2.10 - Arquitectura del Módulo Servicio de Actualización.

Figura 2.11 - Arquitectura del Módulo Aplicación de Administración del servicio de actualización.

La **Capa de Presentación** se encuentra dividida a su vez en dos subcapas: Interfaces de Usuario y Acciones. Las Interfaces de Usuario presentan información a los usuarios y acepta entradas o respuestas de estos para usarlas en el sistema, estas no desarrollan ningún procesamiento de negocio o reglas de validación de negocios. Las Acciones por su parte representan las peticiones que se le hacen al negocio, estas peticiones pueden tener una Interfaz de Usuario asociada o no.

La **Capa de Negocio** es el núcleo del módulo puesto que encapsula la lógica de implementación fundamental para la automatización de los procesos del negocio en cuestión. Esta capa es determinante en el éxito del sistema y su modelación implica especial atención por ambas partes: los desarrolladores y el cliente.

La **Capa de Acceso a Datos** encapsula la lógica de implementación necesaria para gestionar los datos utilizados en uno o muchos procesos de negocio y abstraer así la forma en que los datos persisten o son obtenidos (Tellez, 2008).

Se define una capa transversal que expone el dominio de un módulo conteniendo el conjunto de todas las clases entidades. Las clases entidades son las que encapsulan datos y no el comportamiento de los mismos. Esta capa facilita el paso de información entre distintos componentes o capas que componen a los módulos desde la capa de Acceso a Datos hasta la subcapa de Acciones.

Existen diversos mecanismos para la integración entre capas y todos tienen el objetivo de resolver de una forma u otra un conjunto de problemáticas, tales como:

- Lograr que las capas queden totalmente desacopladas con el objetivo de mejorar la capacidad de mantenimiento de la aplicación, facilitar el desarrollo paralelo y lograr una mayor cohesión de las funcionalidades de la capa.
- Que el desacople no provoque problemas de rendimiento.
- No provocar dependencia a otras tecnologías.
- Facilidad de uso e implementación. Si la integración entre capas requiere un esfuerzo significativo puede provocar atrasos y la posibilidad de cometer errores aumentaría considerablemente (Tellez, 2008).

La solución a estos problemas se propone mediante el uso de varios patrones de diseño que serán aplicados al sistema de actualización propuesto.

2.9.2 Patrones de diseño

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Un sistema informático bien estructurado está lleno de patrones. Cada patrón describe un problema que ocurre una y otra vez en este ambiente, y luego describe el núcleo de la solución a ese problema. Estos patrones se han convertido en la metodología por excelencia de diseño y posterior programación de una solución importante (Larman, 2003).

Los patrones de diseño pretenden:

1. Proporcionar catálogos de elementos reusables en el diseño de sistemas.
2. Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
3. Formalizar un vocabulario común entre diseñadores.
4. Estandarizar el modo en que se realiza el diseño.
5. Facilitar el aprendizaje de las nuevas generaciones de diseñadores concentrando conocimiento ya existente.

En (Larman, 2003) se propone el uso de dos grupos de patrones de diseño, los patrones GRASP[35] y los GoF[36]. Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones, siendo unas de las actividades primordiales en el diseño y construcción de un software. Entre estos patrones GRASP se encuentran: Experto en Información, Creador, Controlador, Alta Cohesión y Bajo Acoplamiento.

Los patrones de diseño GoF son el esqueleto de las soluciones a problemas comunes en el desarrollo de software, se agrupan en tres clasificaciones teniendo en cuenta la función que realicen:

- Creacionales: se encargan de la inicialización y configuración de objetos.
- Estructurales: separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- Comportamiento: describen la comunicación entre objetos o clases.

En el diseño del sistema de actualización del SIGESC fueron utilizados los siguientes patrones GoF:

Patrones de Comportamiento:

- **Command** (Comando): Encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, llevar un registro de las peticiones y poder deshacer operaciones. Las clases de la subcapa de Acciones son un ejemplo del uso de este patrón.

Patrones Estructurales:

- **Facade** (Fachada): Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Cada capa mostrará una interfaz de acople o fachada con el conjunto de funcionalidades que necesite la capa inmediata superior. Este patrón se ve reflejado mediante las clases *ITrackerHosting*, *INAppInstaladas*, *IAccesoDatos*.

Patrones Creacionales:

- **Abstract Factory** (Fábrica Abstracta): En la fachada se definirán fábricas abstractas, por familias de objetos, con el objetivo de definir (no implementar) cómo se van a agrupar los objetos de dicha capa para ser creados por una fábrica concreta que forma parte de la capa en cuestión. La aplicación de este patrón se ve reflejado mediante la clase *FCNegocio, FADAccesoDatos y FCInterfaz*.

[35]GRASP: General Responsibility AssignmentSoftware Patterns, Patrones de Principios Generales de Software para Asignar Responsabilidades.

[36]GOF: Gang of Four, Pandilla de los 4, formada por los ingenieros en sistema Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides.

- **Singleton** (Instancia Única): Está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su objetivo es garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella. Es usado para proveer de instancia única a las clases del negocio *ServicioServidor* y *ServicioCliente* con el objetivo de que las demás clases tengan acceso a las operaciones que en la instancia única se definen.

2.10 Modelo de diseño

En el diseño se modela el sistema para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones de estos. Además impone una estructura del sistema que se debe conservar lo más fielmente posible cuando se dé forma al sistema.

Principales propósitos del diseño:

- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables.

2.10.1 Diagrama de paquetes del diseño

En un paquete de diseño se puede encontrar una colección de clases, relaciones y otros paquetes. Es usado para estructurar el modelo de diseño mediante su división en partes más pequeñas y agrupar elementos relacionados de dicho modelo con propósitos organizacionales (Larman, 2003).

A continuación se muestra la agrupación por paquetes del diseño correspondiente al sistema de actualización.

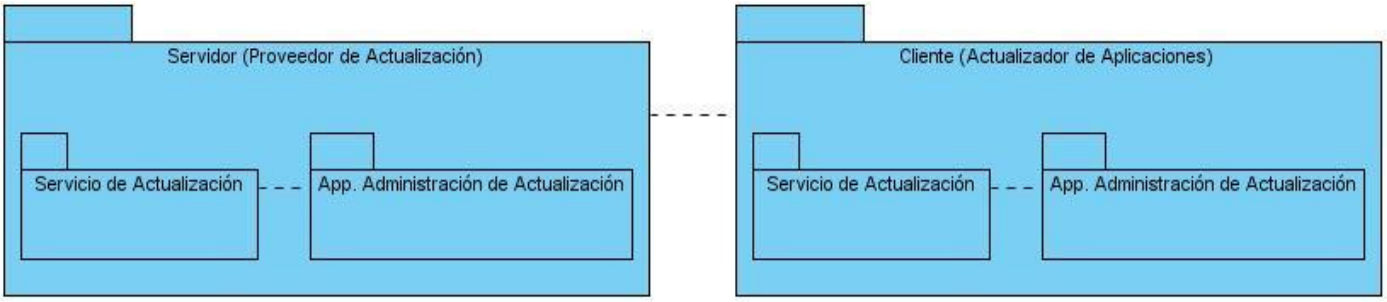

Figura 2.12 Diagrama de Paquetes del Diseño.

2.10.2 Diagrama de clases del diseño

Un diagrama de clases del diseño representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. No todas las clases que aparecían en el modelo del dominio tienen porque aparecer en el diagrama de clases del diseño; solo se incluirán aquellas clases a las que se les ha asignado algún tipo de responsabilidad en el diseño del sistema.

Para una mejor comprensión de los diagramas de clases del diseño (CD) se propone una leyenda con una escala de colores representando las diferentes capas de la aplicación.

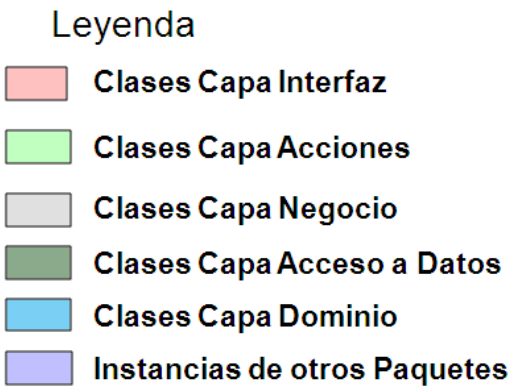

Figura 2.13 Leyenda de las Clases del Diseño.

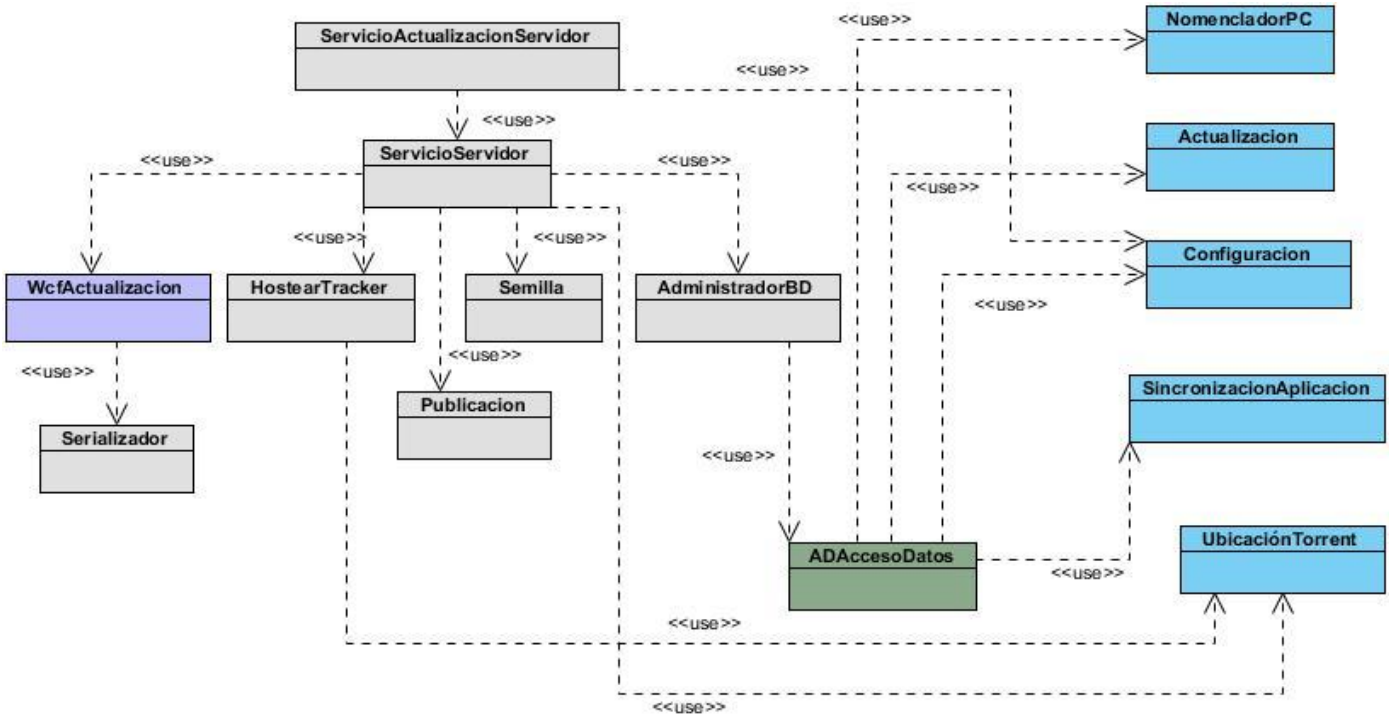

Figura 2.14 Diagrama de CD del Paquete Servidor – Paquete Servicio de Actualización.

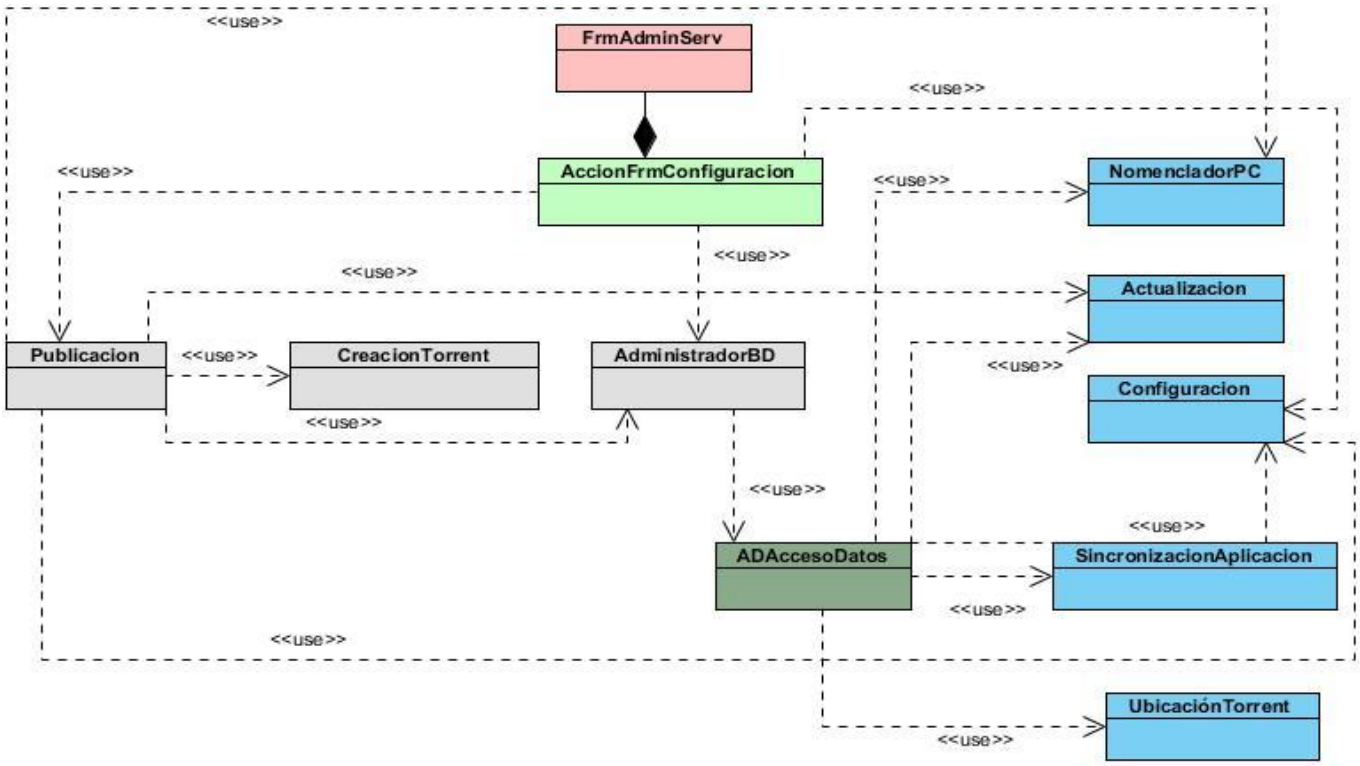

Figura 2.15 Diagrama de CD del Paquete Servidor – Paquete App. Administración de Actualización.

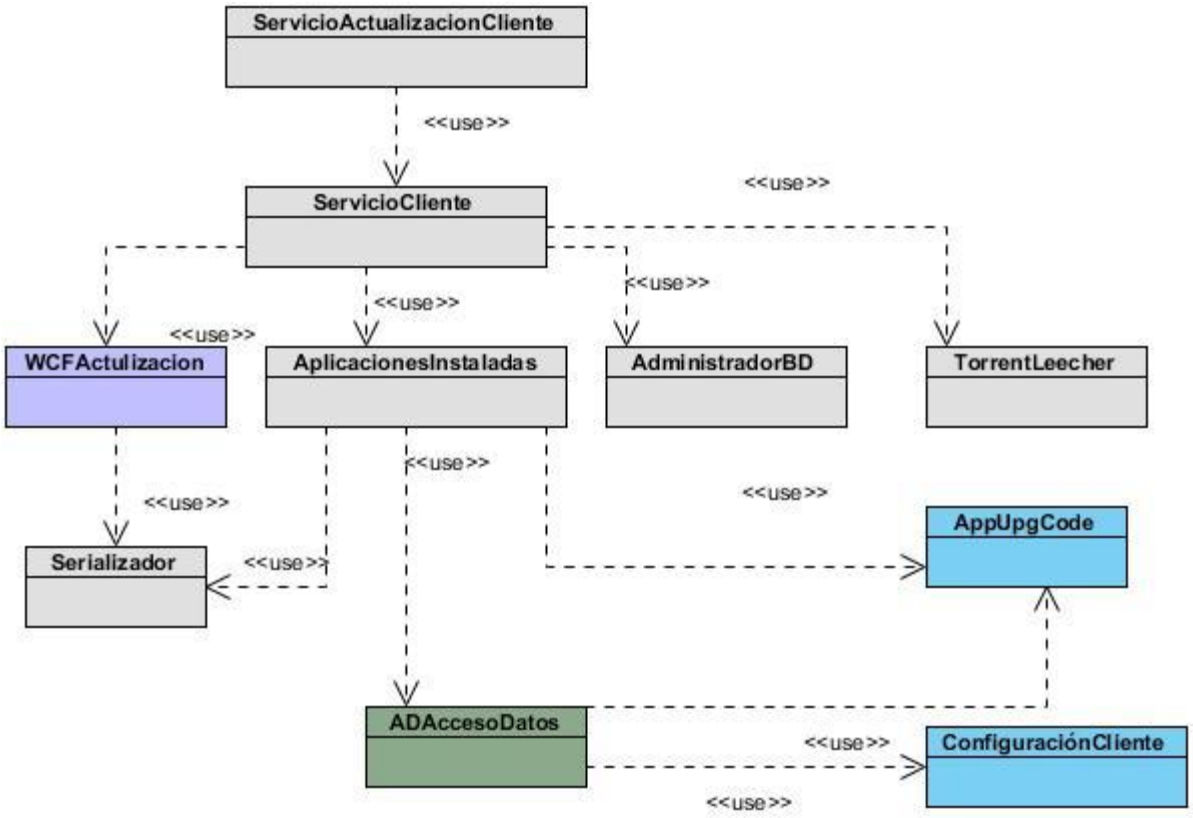

Figura 2.16 Diagrama de CD del Paquete Cliente – Paquete Servicio de Actualización.

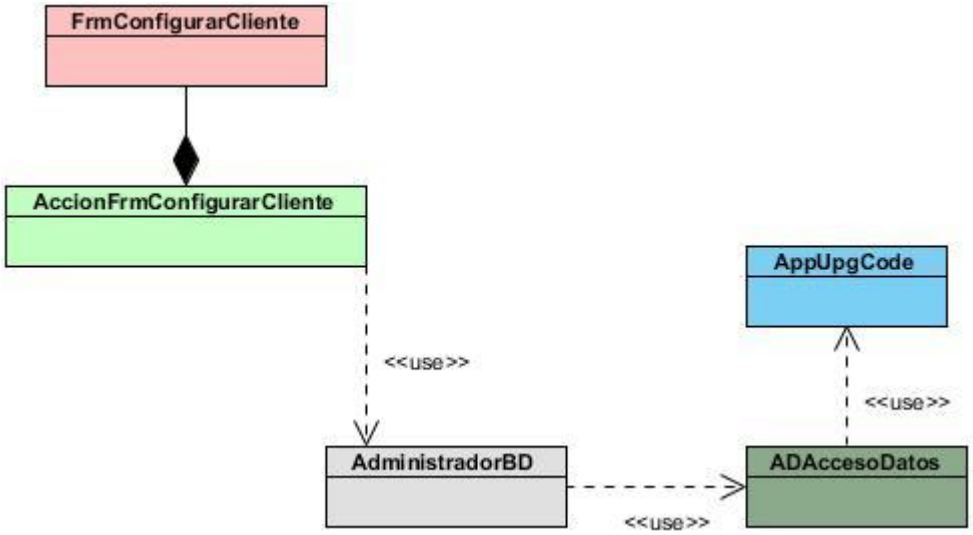

Figura 2.17 Diagrama de CD del Paquete Cliente – Paquete App. Administración de Actualización.

2.11 Diagrama Entidad-Relación

El Diagrama Entidad-Relación (DER) permite que un ingeniero del software identifique objetos de datos y sus relaciones mediante una notación gráfica. En el contexto del análisis estructurado, el DER define

todos los datos que se introducen, se almacenan, se transforman y se producen dentro de una aplicación (Pressman, 2002).

Para la correcta implementación del sistema de actualización es preciso almacenar en una base de datos la información que se maneja en los paquetes de actualización, con el objetivo de que las entidades creadas respondan a las necesidades de las funcionalidades propuestas.

La base de datos se concibió sobre el Gestor de Base de Datos SQLite 3.6, seleccionada porque la información a persistir es pequeña. El motor de SQLite no es un proceso independiente con el que el programa principal se comunica sino que se enlaza con el programa pasando a ser parte integral del mismo. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un solo fichero estándar en la máquina anfitriona (Cabero, 2007). SQLite se ejecuta en muchas plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración. Las transacciones en SQLite tienen la característica de ser atómicas, consistentes, aisladas y durables (ACID) incluso cuando se interrumpen por el fallo del programa, del sistema operativo o de la alimentación del ordenador. Todos los cambios de una transacción en SQLite se hacen completamente o no se hacen.

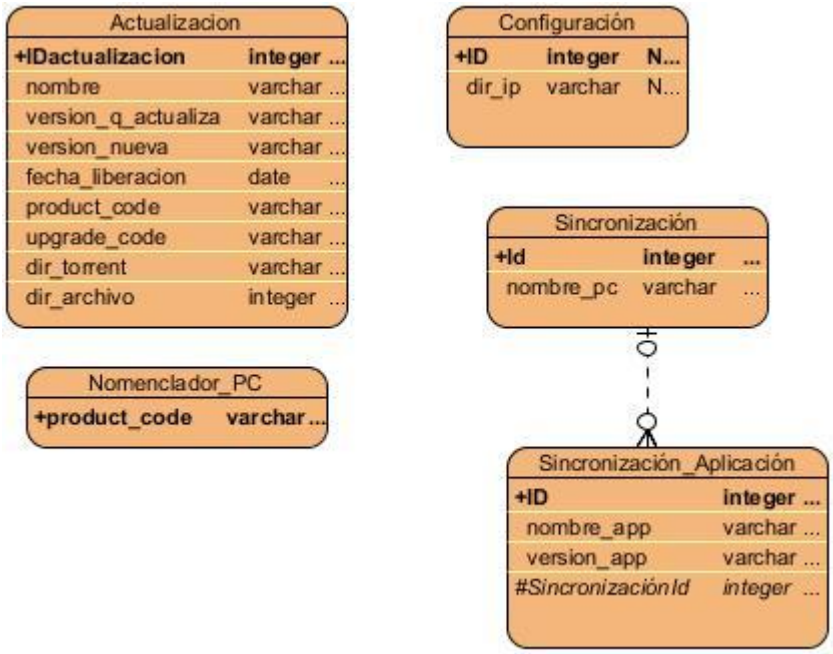

Figura 2.18 Diagrama Entidad-Relación para el Servidor.

Se hace necesario también hacer persistir ciertos datos en los nodos clientes que faciliten el desempeño de las actividades de actualización que se realizan en estaciones independientes por lo que se decide realizar una base de datos ligera, que se enlace a la aplicación cliente. (Ver **Figura 2.19**).

Figura 2.19 Diagrama Entidad-Relación para el Cliente.

2.12 Modelo de implementación

El inicio de la implementación se basa en los resultados obtenidos en el diseño, dichos resultados dan el punto de partida para desarrollar la implementación en términos de componentes como los ejecutables o ficheros de código fuente.

2.12.1 Diagrama de componentes

Los diagramas de componentes modelan la vista estática de un sistema. Tienen un nivel de abstracción más elevado que un diagrama de clases, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, ya que eventualmente un componente puede comprender una gran porción de un sistema (Larman, 2003). El sistema de actualización cuenta con dos aplicaciones que a pesar de que toda la lógica está unida para lograr un solo objetivo y cuentan con la misma arquitectura, son aplicaciones independientes, razón por la cual en el diagrama realizado el término Aplicación se refiere en el lado del “Proveedor de Actualización” a la aplicación servidor y en el “Actualizador de Aplicaciones” se refiere a la aplicación cliente.

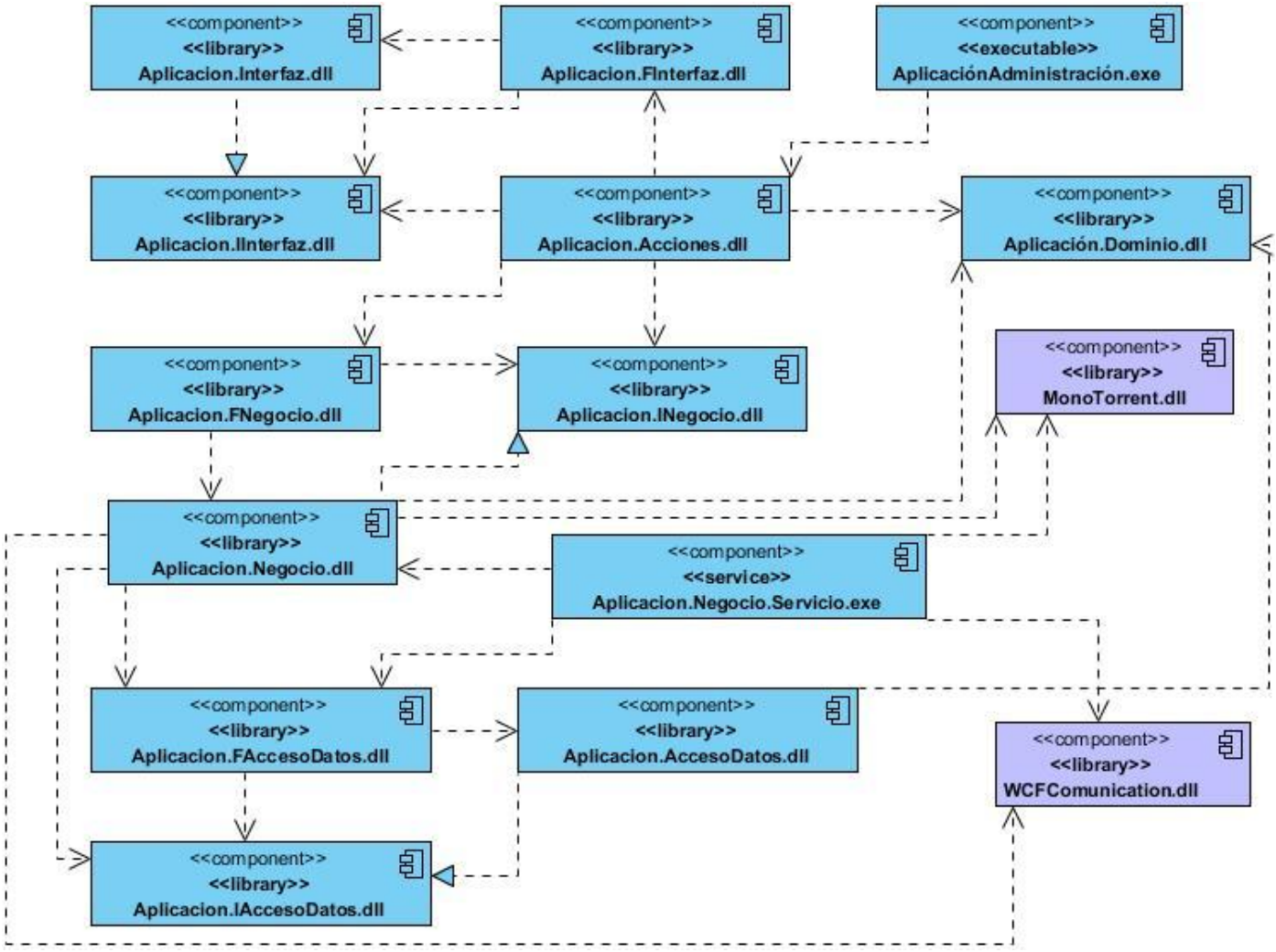

Figura 2.20 Diagrama de componentes del sistema.

2.12.2 Descripción de los componentes

Una mejor descripción de los componentes se encuentra en los anexos.

2.12.2.1 Componente Servidor.AccesoDatos.dll

Propósito

Contiene las clases encargada de gestionar las consultas y modificaciones que se hagan sobre la base de datos.

2.12.2.2 Componente Servidor.Negocio.dll

Propósito

Contiene las clases que controlan el negocio de la aplicación. En estas clases se definen todos los algoritmos o pasos necesarios para obtener un resultado satisfactorio a partir de la petición realizada.

2.12.2.3 Componente Servidor.Negocio.Servicio.exe

Propósito

Es el servidor de actualización. Exactamente es un servicio de Windows cuya función esencial es proporcionar el flujo de tareas que engloba el proceso de actualización desde la aplicación servidor.

2.12.2.4 Componente Servidor.Interfaz.dll

Propósito

Contenedor físico de las clases de interfaz gráfica de usuarios.

2.12.2.5 Componente Servidor.Acciones.dll

Propósito

Contenedor físico de las clases que realizan un flujo de acción para llevar a cabo una operación determinada.

2.12.2.6 Componente Monotorrent.dll

Propósito

Contiene las clases que implementan el protocolo BitTorrent bajo la arquitectura P2P. Es compatible con muchas características avanzadas tales como la encriptación, DHT, el intercambio entre pares, la siembra de contenidos, descarga selectiva y múltiples descargas simultáneas.

2.12.2.7 Componente Servidor.Dominio.dll

Propósito

Contenedor físico de las clases entidades del módulo.

2.12.2.8 Componente AplicaciónAdministración.exe

Propósito

Desencadenar la llamada a la acción que da inicio a la ejecución de la aplicación de administración del servicio de actualización.

2.12.2.9 Componente Cliente.Negocio.Servicio.exe

Propósito

Es el cliente de actualización. Exactamente es un servicio de Windows cuya función esencial es proporcionar el flujo de tareas que engloba el proceso de actualización desde la aplicación cliente.

2.12.2.10 Componente Cliente.IInterfaz.dll

Propósito

Contenedor físico de las clases que definen el comportamiento de los objetos de la capa Interfaz.

2.12.2.11 Componente WCFComunication.dll

Propósito

Es la librería que proporciona las clases que establecen los contratos WCF para lograr la comunicación cliente/servidor a través del canal de conexión TCP.

2.13 Diagrama de despliegue

Un diagrama de despliegue es aquel artefacto encargado de exponer la distribución física de los nodos, los componentes de ejecución y las asociaciones de interacción entre ellos. Los nodos pueden contener instancias de componentes y los componentes pueden contener objetos, indicando la implicación de uno con otro, especificando los tipos de nodos y de componentes (Pressman, 2005).

Para la confección de este modelo se tuvo en cuenta la descripción del mecanismo de actualización expuesto en el epígrafe 2.3, donde la PC_Cliente contiene la aplicación que se desempeña como "Actualizador de Aplicaciones" y el servidor contiene la aplicación que se desempeña como "Proveedor de Actualización" logrando la comunicación entre los nodos por medio de TCP/IP.

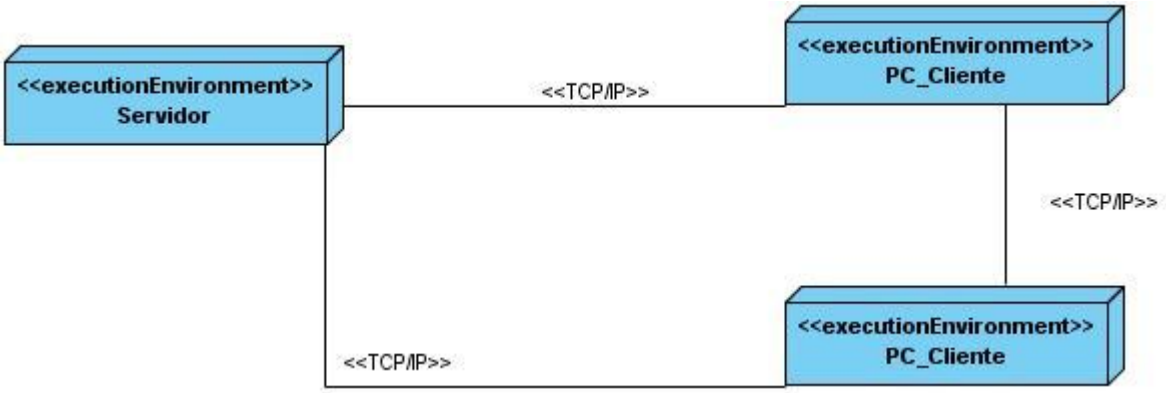

Figura 2.21 Diagrama de Despliegue del sistema.

2.14 Conclusiones

En el capítulo se describe el estudio realizado de los procesos del negocio que originan los problemas y deficiencias del entorno analizado. Con los resultados obtenidos se desarrolló una propuesta basada en un modelo híbrido con el propósito de solucionar las carencias e inconvenientes existentes en el mecanismo de actualización del SIGESC.

Posterior al análisis referente a las características del diseño propuesto, las necesidades expuestas y un estudio sobre las arquitecturas de software realizado se concretó la arquitectura que guiaría el diseño y la implementación del componente mediante la composición de tres estilos arquitectónicos: Cliente-Servidor, Punto a Punto y tres capas para la implementación. La unión de estos estilos conforma una arquitectura concreta que soluciona las necesidades y respalda la implementación posterior. Paralelamente con la elección de los patrones de diseño apropiados se alcanza un desarrollo organizado y un producto de alta calidad.

Se elaboró el modelo de implementación del sistema de actualización, garantizando la implementación correcta de los patrones de diseño propuestos y de la arquitectura de tres capas definida. Se definió finalmente la forma en la que se desplegará el sistema según el mecanismo de actualización presentado.

Luego de culminada la fase de diseño e implementación se procede a realizar las pruebas al sistema propuesto, con el objetivo de verificar el cumplimiento de los requisitos captados y la calidad del software.

Capítulo III: Validación de la solución

3.1 Introducción

En el presente capítulo se expone el modelo de pruebas, con el cual se define una forma de comprobar las funcionalidades principales de la solución planteada y el cumplimiento del objetivo general de esta investigación, para lo que se propone el uso de la prueba de caja negra y pruebas de rendimiento al sistema, con el propósito de validar la solución propuesta y encontrar fallas no detectadas hasta entonces. El flujo de pruebas es muy importante en la Ingeniería de Software ya que es una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, posteriormente una evaluación es hecha de algún aspecto del sistema o componente.

3.2 Pruebas

La prueba de software es un elemento crítico para garantizar la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación, la prueba se enfoca principalmente en la evaluación y determinación de la calidad del producto.

Las pruebas evalúan el producto y permiten determinar si cumple con el objetivo previsto, por lo que es necesario diseñar un plan de pruebas que se adapte y sea coherente con la metodología de desarrollo, que proporcione un enfoque de fácil acceso a la estructura para verificar los requisitos y cuantificar su rendimiento, y que identifique las diferencias entre los resultados previstos y los reales. Es el proceso por medio del cual, se evalúa la correcta interpretación y aplicación de los requisitos especificados.

3.3 Pruebas de Caja Negra

Las pruebas de caja negra, también denominadas pruebas de comportamiento, permiten estudiar las entradas, las salidas o respuestas que se producen en un sistema sin tener presenta su funcionamiento interno. Permite la validación y verificación de la interfaz de usuario, entendiendo por interfaz las entradas y salidas de dicho programa, donde primeramente se revisa el modelo de la interfaz para garantizar que se ajusta a los requisitos y otros elementos que puedan haber surgido durante el análisis, se tienen en cuenta los aspectos específicos de la aplicación en su interacción con los usuarios. Una de las ventajas que brinda el proceso de las pruebas es el valor agregado que le añaden al producto sobre la facilidad de uso del sistema (Pressman, 2005).

El objetivo fundamental de este tipo de prueba es comprobar que cada función del software es operativa. En la preparación de los casos de prueba se necesita conocer un grupo de datos que determinarán como

se ejecutará cada uno de los casos y a través de los cuales el sistema se pueda ejecutar en todas sus variantes: datos válidos o no válidos si lo que se desea probar es un error o una funcionalidad (Pressman, 2005). Por lo que estos datos se determinan atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que este se ejecute correctamente. Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

1. Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
2. Técnica del Análisis de Valores Límites: esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
3. Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones (Pressman, 2005).

El método a utilizar en la aplicación de las pruebas de caja negra, es el de partición equivalente, pues reduce el número total de casos de prueba a desarrollar, obteniéndose así un conjunto manejable que evalúe el software.

3.3.1 Técnica de la Partición de Equivalencia

Una partición equivalente es una técnica de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada (Pressman, 2005).

El diseño de casos de prueba según esta técnica consta de dos pasos:

1. Identificar las clases de equivalencia.
2. Identificar los casos de prueba.

Con la aplicación de esa técnica se obtiene un conjunto de pruebas que reduce el número de casos de pruebas e indican sobre la presencia o ausencia de errores.

3.4 Descripción de los casos de prueba

Se seleccionaron dos casos de prueba que representan las funcionalidades más significativas en la solución desarrollada, con el objetivo de verificar:

- Que los resultados esperados ocurran cuando se usen datos o entradas válidos.
- Que sean desplegados los mensajes apropiados de error y precaución cuando se usan datos o entradas inválidos.
- Funcionalidades incorrectas o ausentes.

- Errores de interfaz.

3.4.1 Caso de prueba: Publicar Actualización

Escenario (EC)	Descripción	Flujo central
EC 1 Publicar Actualización.	Se publican las actualizaciones en la ubicación seleccionada por usuario.	1. Seleccionar la opción “Publicación“. 2. Selecciona la opción “Examinar” para seleccionar los paquetes de actualización a publicar. 3. Introducir la ubicación del repositorio donde se desea que deban estar ubicados los paquetes de actualización. 4. Seleccionar la opción “Publicar”.
EC 2 Datos Obligatorios Nulos.	Se indica que los datos no pueden ser nulos.	1. Seleccionar la opción “Publicación “. 2. Selecciona la opción “Examinar” para buscar la ubicación a publicar. 3. Selecciona la ubicación de los paquetes de actualización. 4. Seleccionar la opción “Publicar Actualización” para publicar la información. 5. Seleccionar la opción “Aceptar” para salir del mensaje.
EC 3 Dirección No Válida	Se indica que la dirección a publicar no es válida.	
EC 4 Cerrar	Se cierra la interfaz del Proveedor de Actualización.	1. Seleccionar la opción “Publicación“. 2. Selecciona la opción “Cerrar”.
EC 5 Eliminar	Se elimina del componente y de la lista el paquete de actualización seleccionado.	1. Seleccionar el paquete de actualización a eliminar. 2. Selecciona la opción “Eliminar”.

Id del Escenario	EC 1	EC 2	EC 3	EC 4
Escenario	Publicar Actualización.	Datos Obligatorios Nulos.	Dirección IP No Válida	Cerrar

Respuesta del Sistema.	El sistema lee las propiedades de los paquetes de actualización que fueron publicados, guarda en la base de datos estas propiedades y crea los archivos Torrent por cada paquetes de actualización publicados en una ubicación con el siguiente formato: *Disco Local (Ej. C): Archivos Comunes\ActualizacionSIGESC\ Version_#.*	Muestra el siguiente mensaje: "Dirección IP vacía".	Muestra un mensaje indicando que en la ubicación seleccionada no se encuentran paquetes de actualización.	Cierra la interfaz del Proveedor de Actualización.

3.4.2 Caso de Prueba: Actualizar aplicaciones

Escenario	Descripción	Flujo central
EC 1 Actualización exitosa.	Se inician las tareas que engloban la actualización de las aplicaciones del SIGESC de una estación programadas por el usuario cada cierto intervalo de tiempo.	1. Comprobar las aplicaciones del SIGESC instaladas y la versión de cada una en la estación. 2. Conectar a la dirección IP del servidor de actualizaciones configurado. 3. Verificar si existen actualizaciones para los productos instalados. 4. Descargar paquetes de actualización. 5. Eliminar los paquetes de actualización descargados en el directorio de la estación.
EC 2 Aplicaciones no Instaladas	Se indica que no existen aplicaciones del SIGESC instaladas en la estación de trabajo.	1. Comprobar las aplicaciones del SIGESC instaladas y la versión de cada una en la estación.
EC 3 Fallo de Conexión.	Se indica que hubo un fallo de conexión.	1. Comprobar las aplicaciones del SIGESC instaladas y la versión de cada una en la estación.

		2. Conectar a la dirección IP del servidor de actualizaciones configurado.
EC 4 No encontró actualizaciones	Se indica que no existen actualizaciones para los productos instalados.	1. Comprobar las aplicaciones del SIGESC instaladas y la versión de cada una en la estación. 2. Conectar a la dirección IP del servidor de actualizaciones configurado. 3. Verificar si existen actualizaciones para los productos instalados.

Id del Escenario	EC 1	EC 2
Escenario	Actualización exitosa.	Aplicaciones no Instaladas.
Respuesta del Sistema.	El sistema comprueba qué aplicaciones del SIGESC se encuentran instaladas en la estación, verifica en el servidor de actualizaciones si se han publicado actualizaciones superiores a las instaladas en la estación, descarga el o los archivos Torrent y a través de este se comienzan a descargar los paquetes de actualización, comprueba que las aplicaciones a las que se va a actualizar no estén ejecutándose en la estación y se instalan las actualizaciones.	Guarda en un Log del sistema el error de la inexistencia de aplicaciones del SIGESC instaladas en la estación.

Id del Escenario	EC 3	EC 4
Escenario	Fallo de Conexión.	No encontró actualizaciones

Respuesta del Sistema.	Guarda en un Log del sistema el error de que hubo un fallo de conexión con el servidor de actualizaciones.	Guarda en un Log del sistema el error de que no se encontraron actualizaciones para las aplicaciones.

3.4.3 Resultado de las Pruebas

Los casos de pruebas ejecutados arrojaron resultados satisfactorios, demostrando que el sistema desarrollado realiza todas las funcionalidades previstas, las cuales logran realizar un proceso de actualización acotado a las necesidades del SIGESC, también permitieron descubrir errores en las interfaces y mejorar el rendimiento del sistema.

3.5 Pruebas de Rendimiento

Las pruebas de rendimiento permiten determinar lo rápido que realiza una tarea un sistema informático en condiciones particulares de trabajo. Los resultados permiten analizar para un conjunto de datos de entrada y un tiempo determinado, el comportamiento del sistema.

Las pruebas se realizaron en el laboratorio de proyecto donde se simularon los distintos escenarios por los que puede atravesar el mecanismo de actualización del SIGESC, con el fin de validar el cumplimiento de uno de los objetivos principales de la investigación en cuanto a la agilidad de las tareas de distribución de archivos, dado que en el SIGESC, como sistema que se despliega en distintos estados de la República Bolivariana de Venezuela, mantener todas las aplicaciones que lo conforman actualizadas con las últimas versiones puede llevar mucho tiempo y recursos.

El entorno en que se realizaron las pruebas se destaca por la presencia de las siguientes características:

Hardware	**Datos**
Procesador	Intel(R) Celeron(R)CPU 420 a 1.60 GHz (Cliente) Core 2 Duo a 2.2 GHz(Servidor)
Memoria	512 MB(Cliente) 2 GB (Servidor)
Características de la red	**Datos**
Topología de la red	Árbol Jerárquico
Ancho de banda	100 Mbps
Software	**Datos**
Sistema Operativo	Windows XP SP 3 (Cliente-Servidores) Windows 7 Ultimate (Servidor)
Antivirus	Kaspersky Workstation 6.0 (Cliente- Servidor)

Los escenarios en los que fue probado el sistema de actualización se destacan por las siguientes características y resultados:

Escenario Efectuado	Resultado de la Prueba
Existencia de un proveedor de actualizaciones central y cuatro clientes de actualización.	Los clientes se configuraron para encuestar al servidor cada cinco minutos, lo cual arrojó que la comprobación, descarga e instalación de las actualizaciones ocurrieran en un tiempo de un minuto.
Existencia de un proveedor de actualizaciones central, un proveedor de actualizaciones intermediario y tres estaciones de trabajo con el actualizador de aplicaciones instalado.	Los clientes se configuraron para encuestar al proveedor intermediario cada tres minutos y este se configuró para encuestar al primer proveedor cada cinco minutos, lo cual arrojó que la comprobación, descarga e instalación en los clientes finales ocurrieran en dos minutos, siempre teniendo en cuenta la necesidad de la actualización de las aplicaciones y de los paquetes de actualización previstos para la prueba realizada.
Existencia de un proveedor de actualizaciones central, dos proveedores de actualizaciones intermediario donde uno descarga del otro y a su vez el último provee actualizaciones a tres estaciones de trabajo.	Los clientes situados en las estaciones de trabajo se configuraron para encuestar a un proveedor de actualizaciones cada cinco minutos, este último a su vez se configuró para encuestar a otro proveedor de actualizaciones cada tres minutos y este a su vez se configuró para descargar de un proveedor central de actualizaciones cada cinco minutos. La prueba arrojó que todo el proceso se realizara en el tiempo de tres minutos.

Los resultados arrojados por estas pruebas constataron que la solución ejecuta correctamente todas las funcionalidades previstas. El rendimiento de los servicios de actualización tanto para el cliente como para el servidor cumple con las necesidades del SIGESC y corrobora con las características requeridas para el despliegue de la solución.

3.6 Conclusiones

En este capítulo se observó la importancia del flujo de pruebas en la Ingeniería de Software, destacándose los beneficios que reporta este proceder al desarrollo de cualquier sistema, ya que las pruebas se enfocan principalmente en la evaluación y determinación de la calidad del producto. Finalmente se seleccionó la prueba de caja negra con la cual se pueden estudiar las entradas, las salidas o respuestas que se producen en un sistema sin tener presente su funcionamiento interno. Además se mencionaron las diferentes técnicas existentes dentro de la misma, escogiéndose para ser utilizada la Técnica de la Partición de Equivalencia, la cual divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Para la aplicación de la técnica se construyeron casos de prueba basados en las clases de equivalencia, logrando determinar un producto funcional acorde con las tendencias actuales y que responde a la problemática de la investigación desarrollada. Se realizaron pruebas de rendimiento a la aplicación que arrojaron un resultado positivo pues el sistema logra realizar la distribución de los paquetes de actualización en un tiempo considerablemente menor al que se usa en el mecanismo de actualización existente.

Conclusiones Generales

Con la implantación del mecanismo de actualización propuesto y el desarrollo de un **Sistema de Actualización para el Sistema de Gestión de Emergencias de la Seguridad Ciudadana** se obtienen los siguientes beneficios:

- El estudio desarrollado para analizar las características de los sistemas de actualizaciones automáticas, permitió conformar una base teórica que agrupó los conocimientos para solucionar la problemática planteada, lo que permitió a elegir las tecnologías y herramientas candidatas para la construcción del sistema informático.
- El diseño propuesto reunió los elementos apropiados para un desarrollo organizado que cumple con las necesidades y los estándares referenciados.
- Los resultados de la implementación arrojaron la obtención de un sistema compuesto por dos aplicaciones para la actualización automática de las aplicaciones del SIGESC, que al mismo tiempo aporta ventajas con respecto al despliegue, mantenimiento y utilización.
- Las pruebas de caja negra realizadas al software validaron la correcta implementación de las funcionalidades requeridas, demostrando buen desempeño de la ejecución del sistema.
- El tiempo utilizado para el proceso de actualización disminuye considerablemente comparado con el método que se utiliza de actualización manual, donde en este último el proceso puede demorar más de un mes en dependencia de los recursos económicos que se necesiten emplear para este y de la disponibilidad del equipo de soporte; por lo tanto con el sistema de actualización desplegado en un entorno de prueba y considerando que no existan problemas ajenos a la solución en cuanto las conexiones de red entre las computadoras, la duración de estas fue mínima, considerando que el servicio de actualización puede ser configurado para iniciar las tareas cada cierto tiempo.

Se arriba a la conclusión de que el objetivo principal de este trabajo se ha materializado con la realización de las tareas investigativas planteadas.

Recomendaciones

Tras haber cumplido el objetivo general del presente trabajo de diploma se recomiendan los siguientes aspectos:

- Aplicar los conocimientos adquiridos durante el transcurso de esta investigación en otros trabajos enmarcados en la misma línea investigativa.
- Estudiar la viabilidad de aplicar el mecanismo propuesto en otros proyectos de la Universidad.
- Desplegar el Sistema de Actualización en un escenario real donde estén presentes las aplicaciones del SIGESC.
- Investigar sobre nuevas tecnologías similares a BitTorrent que hagan una mejor gestión del ancho de banda.
- Adicionar una interfaz gráfica para controlar el estado de las descargas y de las conexiones entre el servidor y el cliente.

Glosario de Términos

Abreviaturas	**Significado**
Aplicación (sistema)	Sistema que ofrece a un usuario final un conjunto coherente de casos de uso.
Arquitectura Cliente-Servidor	Este modelo arquitectónico define dos componentes esenciales, el primero que es el ente que ofrece un servicio, el servidor y el segundo que demanda el servicio, el cliente. En este modelo la lógica del sistema es separada en estos dos elementos, garantizando que los procesos sean independientes pero que a su vez funcionen de forma cooperativa para obtener los resultados requeridos.
Arquitectura Punto a Punto (P2P)	Se caracteriza fundamentalmente porque su funcionamiento básico es distribuido y descentralizado, es decir, no existe la presencia de un servidor central que gestione peticiones, en cambio, cada componente puede laborar con el rol de cliente o de servidor balanceando la carga y los servicios entre todos los elementos del modelo.
BitTorrent	Protocolo diseñado para el intercambio de ficheros de igual a igual.
C#	Lenguaje de Programación (C - Sharp).
CASE	Ingeniería de Software Asistida por Ordenador. *Computer Aided Software Engineering*.
DLL	Biblioteca de vínculos dinámicos (*Dynamic-Link Library*).
Entidades	Representa un contenedor de información, algo físico que se utilice en el proceso del negocio y que

	sirva para obtener información o para actualizar información. Generalmente tiene estados, en dependencia de en qué momento aparezca como parte del proceso.
Framework	Es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y un lenguaje de *scripting* entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
MSI	Paquetes de software que contienen la información necesaria para automatizar una instalación sin necesidad de intervención manual del usuario.
MSP	Son los paquetes de revisión de software; estos archivos se pueden distribuir como parches o actualizaciones para solución de problemas. Las revisiones no deben usarse para cambios importantes y sus efectos están limitados.
Parches	En informática, se refiere a cambios que se aplican a un programa, para corregir errores, agregarle funcionalidad o actualizarlo.
Protocolo de Control de Trasmisión (TCP)	Permite a dos anfitriones establecer una conexión e intercambiar datos. El TCP garantiza la entrega de datos, es decir, que los datos no se pierdan durante la transmisión y también garantiza que los paquetes sean entregados en el mismo orden en el cual fueron enviados.
Protocolo de Internet (IP)	Protocolo no orientado a conexión, usado tanto por el origen como por el destino para la comunicación de datos, a través de una red de paquetes conmutados no fiable y de mejor entrega posible

	sin garantías.
WCF	Windows Communication Foundation
WSUS	Windows Server Update Service, servicio de actualización de Windows.

Referencias Bibliográficas

1. **Ajmani, Sameer. 2004.** *Automatic Software Upgrades for Distributed Systems.* Massachusetts : s.n., 2004.

2. **BitTorrent. 2012.** BitTorrent. [En línea] 2012. [Citado el: 2012 de Enero de 10.] http://www.bittorrent.com.

3. **Cabai, Diego. 2008.** Configurando el ancho de banda de WSUS con BITS. *Diego Cabai Blog.* [En línea] 8 de Julio de 2008. [Citado el: 4 de Febrero de 2012.] http://www.cabai.com.ar/2008/07/configurando-el-ancho-de-banda-del-wsus-con-bits.html.

4. **Cabero, Gerardo Antonio. 2007.** *Sqlite: Rápido, ágil,liviano y robusto.* 2007.

5. **Calderón, Alberto. 2006.** Alberto Calderón. *El Protocolo BitTorrent.* [En línea] 27 de Agosto de 2006. [Citado el: 5 de Diciembre de 2011.] http://www.albertocalderon.cl/web-juegos-y-otros/el-protocolo-bittorrent.

6. **Campo, Adonis Cesar Legón. 2010.** *Sistema de Actualización Automática .* Ciudad de la Habana : UCI, 2010.

7. **Cisco Networking Academy. 2011.** CAPITULO 3 Protocolos y funcionalidad de la capa de Aplicación. *CCNA Exploration 4.0 Aspectos básicos de Networking.* USA : Scribd, 2011.

8. **Dumitras, Tudor. 2009.** *Toward Upgrades-as-a-Service in Distributed Systems.* 2009.

9. **Durell, W.R. 2005.** *Data Administration. A Practical Guide to Data Administration.* 2005.

10. **Elhajj. 2005.** *Guía paso a paso para empezar a trabajar con Microsoft Windows Server Update Services.* s.l. : Sean Bentley, 2005.

11. **Fernández, Pedro Armando Pérez. 2010.** *Infraestructura para el soporte de telemedicina en aplicaciones Neuronic.* Ciudad de la Habana : Universidad de la Habana, 2010.

12. **Frank. 2010.** Gigle.net. *El uso de tecnología BitTorrent hace los procesos entre servidores de Twitter 75 veces más rápidos.* [En línea] 17 de Julio de 2010. [Citado el: 7 de Febrero de 2012.]

http://www.gigle.net/el-uso-de-tecnologia-bittorrent-hace-los-procesos-entre-servidores-de-twitter-75-veces-mas-rapidos/.

13. **García, Gilberto Pedraza. 2008.** *Evolución e Integración de Aplicaciones Legadas: Comenzar de Nuevo o Actualizar?* Bogotá : Universidad Piloto de Colombia, 2008.

14. **IEEE. 2000.***IEEE Std 1471-2000.* 2000.

15. **Jansen, Slinger y Ballintijn, Gerco. 2004.** *A Process Model and Typology for Software Product Updaters.* Amsterdam, The Netherlands : s.n., 2004.

16. **Jeff. 2008.** Kioskea.net. *Alta Disponibilidad.* [En línea] 16 de Ocutbre de 2008. [Citado el: 21 de Febrero de 2012.] http://es.kioskea.net/contents/surete-fonctionnement/haute-disponibilite.php3

17. **Kaspersky Lab. 2009.** *Kaspersky Administration Kit 8.0.* 2009.

18. **Kaspersky Lab. 2011.** Kaspersky Lab Technical Support. *Agente de actualización.* [En línea] 19 de Agosto de 2011. [Citado el: 10 de 12 de 2011.] http://support.kaspersky.com/sp/ak8/update?qid=208280730. 2707.

19. **Keating, Tom. 2010.** TMCnet Bloggers. *Facebook Uses BitTorrent to Upgrade Servers.* [En línea] 25 de Junio de 2010. [Citado el: 06 de Febrero de 2012.] http://blog.tmcnet.com/blog/tom-keating/social-networking/facebook-uses-bittorrent-to-upgrade-servers.asp.

20. **Larman, Craig. 2003.** *UML y Patrones.* 2003.

21. **Makey, Alex. 2010.** *Introducing .Net 4.0 wiht Visual studio 2010.* New York, United States of America : Springer-Verlag New York, Inc., 2010.

22. **Microsoft. 2003.** *Cómo Implantar Microsoft Software Update Services.* USA : Microsoft Corporation, 2003.

23. **Microsoft Corporation. 2005.** Microsoft Patterns and Practices. *Introduction to the Updater Application Block.* [En línea] 5 de 2005. [Citado el: 2 de 11 de 2011.] http://msdn.microsoft.com/en-us/library/ff648675.aspx.

24. **Microsoft. 2007.** MSDN. [En línea] 2007. [Citado el: 18 de Febrero de 2012.] http://msdn.microsoft.com/es-es/library/h2zwd6bw%28v=vs.90%29.aspx.

25. **Microsoft. 2012.** Microsoft System Center. [En línea] 2012. [Citado el: 10 de Febrero de 2012.] http://www.microsoft.com/systemcenter/es/es/default.aspx.

26. **Microsoft Technet. 2011.** Microsoft Technet. *Configurar Servicios de Archivo.* [En línea] 2011. [Citado el: 28 de 11 de 2011.] http://technet.microsoft.com/es-es/library/dd567676.aspx.

27. **Microsoft. 2008.** *Microsoft® Deployment Toolkit 2008 Deployment Concepts.* s.l. : Solution Accelerators, 2008.

28. **Norton. 2008.** Norton By Symantec. *Por qué son vitales las actualizaciones de seguridad.* [En línea] 1 de 8 de 2008. [Citado el: 15 de 10 de 2011.] http://mx.norton.com/products/library/article.jsp?aid=vital_security#actualizaciones..

29. **Pardo Tapia, Alejandro y Martínez Campos, Diego. 2010.** *Protocolo BitTorrent.* s.l. : UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA, 2010.

30. **Pérez, Oscar Alberto Gallardo. 2004.** Universidad Francisco de Paula Santander. [En línea] 2004. [Citado el: 04 de Mayo de 2012.] http://bari.ufps.edu.co/materias/proyso/htm_docs/pag_smon.htm.

31. **Pérez, Roger. 2003.** *Programacion Orientada a Objetos con C#.* Matanzas, Cuba : s.n., 2003.

32. **Pressman, R.S. 2002.** *Ingeniería de Software. Un enfoque práctico.* 2002.

33. **Pressman, Roger. 2005.** *Ingeniería del Software. Un enfoque práctico.* 2005. 6ta Edición.

34. **Reinier Quevedo Batista, Laritza Laguardia López. 2011.** *Infraestructura de Actualización para el Sistema de Información Hospitalaria alas HIS.* Ciudad de la Habana : UCI, 2011.

35. **Sharp, John. 2007.** *Microsoft Windows Communication Foundation: Step bye Step.* United States of America : Microsoft Press, A Division of Microsoft Corporation, 2007.

36. **SQLite.org. 2000.** SQLite. *SQLite.org.* [En línea] 2000. [Citado el: 04 de Abril de 2012.] http://www.sqlite.org/index.html.

37. **System Center. 2011.** Microsoft System Center. *System Center Updates Publisher.* [En línea] 2011. [Citado el: 20 de Febrero de 2012.] http://technet.microsoft.com/es-es/library/bb531022.aspx.

38. **Tellez, Eddy Sánchez. 2008.** *Especificación de la Arquitectura Base del Sistema de Gestión de Emergencia y Seguridad Ciudadana 171.* Ciudad de la Habana : UCI, 2008.

39. **Victoria. 2007.** Definición ABC. *Definición de Fichero.* [En línea] 2007. [Citado el: 28 de 11 de 2011.] http://www.definicionabc.com/general/fichero.php.

Bibliografía

1. **Blasi, Emanuel. 2005.** *Resumen de Patrones de Diseño.* Buenos Aires, Argentina : s.n., 2005.
2. **Campo, Adonis Cesar Legón. 2009.** *Servicio y componentes para la actualización automática de aplicaciones y recursos.* Ciudad de la Habana : s.n., 2009.
3. **Elhajj, Tim. 2005.** *Deploying Microsoft Windows Server Update Services.* 2005.
4. **Gavin Bierman, Michael Hicks, Peter Sewell, Gareth Stoyle. 2003.** *Formalizing Dynamic Software Updating.* Estados Unidos : s.n., 2003.
5. **Ingris Valdés Pérez, Eberto Cepero Abreu. 2011.** *Componente genérico para la comunicación en sistemas distribuidos.* Ciudad de la Habana, Cuba : UCI, 2011.
6. **John F. Buford, Heather Yu,Eng Keong Lua. 2009.** *P2P: Network and Applications.* Estados Unidos : Elsevier Inc, 2009.
7. **Pablo Cibraro, Kurt Claeys. 2010.** *Professional WCF 4: Windows Communication Foundation with .NET 4.* Canada : Wiley Publishing, Inc., Indianapolis, Indiana, 2010.
8. **Rolando Alfredo Hernández León, Sayda Coello González. 2011.** *El proceso de Investigación Científica.* Ciudad de la Habana : Editorial Universitaria del Ministerio de Educación Superior, 2011.
9. **S.Somasegar, David Hill, Scott Guthrie. 2009.** *Microsoft Application Arquitecture Guide 2da Edición.* Estados Unidos : s.n., 2009.
10. **Steve Resnick, Richard Crane, Chris Bowen. 2008.** *Essential Windows Comunication Foundation For .Net Framework 3.5.* Estados Unidos : Pearson Education, Inc, 2008.
11. **César De la Torre Llorente, Miguel Ángel Ramos Barroso. 2010.** *Guía de Arquitectura N-Capas Orientada al Dominio con .Net 4.0.* España: Microsoft Ibérica S.R.L, 2010.
12. **Tellez, Eddy Sánchez. 2008.** *Especificación de la Arquitectura Base del Sistema de Gestión de Emergencia y Seguridad Ciudadana 171.* Ciudad de la Habana : UCI, 2008.
13. **Giuffrida, Cristiano. 2009.** Cooperative Update: A New Model for Dependable Live Update. Orlando, Florida.