Universidad de las Ciencias Informáticas

Facultad 3

TÍTULO:

Propuesta de un componente de actualización para las aplicaciones clientes del Sistema de Informatización del Registro de la Cámara de Comercio

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Autores: Yasiel Dueñas Sánchez

Carlos Ernesto del Junco La Rosa

Tutor: Ing. Julio Cesar Prieto Álvarez

La Habana 2013

"Año 55 de la Revolución"

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Autores:

_____ _____

Yasiel Dueña Sánchez Carlos Ernesto del Junco La Rosa

Tutor:

_____

Ing. Julio Cesar Prieto Álvarez

Resumen

Las actualizaciones de los sistemas informáticos forman parte de su evolución y constante desarrollo. Estas actualizaciones constituyen el mecanismo por excelencia para brindar su permanencia y funcionalidad a lo largo del tiempo. De ahí que una solución informática que sea empleada, es muy probable que sea actualizada. El Sistema de Informatización del Registro de la Cámara de Comercio de la República de Cuba (SIRECC) perteneciente al Centro de Gobierno Electrónico (CEGEL) no cuenta con un proceso para llevar a cabo sus actualizaciones de forma automática. En el presente trabajo de diploma se desarrolla un componente para realizar de manera automática dichas actualizaciones en las estaciones clientes de la Cámara de Comercio de la República de Cuba. Mostrándose la metodología y los principales artefactos utilizados en su construcción, así como las diferentes pruebas aplicadas para su validación. Contribuyendo así este componente en la disminución del tiempo de ejecución del proceso de actualización y la diferencia de versiones de los sistemas.

Índice

Tablas

Figuras

Ecuaciones

Introducción

Con el creciente avance de la ciencia y la tecnología la informática ha evolucionado notablemente, de tal forma que ha logrado convertirse en uno de los medios fundamentales para el desarrollo tecnológico de la sociedad. Este adelanto tecnológico combinado con el auge de las ciencias informáticas, ha propiciado que la complejidad de los proyectos de desarrollo de software se haya incrementado de manera considerable.

De igual forma las actividades comerciales han sido un factor de vital importancia para el desarrollo de las sociedades a nivel mundial. De ahí la intención de establecer una institución social, como organismo representativo para regular el comercio, las denominadas cámaras de comercio. Las cuales son creadas como instituciones o asociaciones que operan en determinada región de las divisiones político – administrativas, y que tienen como función proteger los intereses de los comerciantes que desarrollan actividades industriales, de importación, exportación, mayoristas, minoristas y otras.

Por su parte la Cámara de Comercio de la República de Cuba, es una asociación de empresas vinculadas al comercio, la industria y los servicios, con reconocimiento ante los organismos del estado, la que permite orientar mejores alternativas para el desarrollo de la actividad empresarial de las entidades que la integran. Además constituye un mecanismo para la reinserción de la economía cubana en el mundo de las relaciones económicas internacionales, pues potencia e intercambia información valiosa en torno a las posibilidades de negocios a escala mundial. (1)

En el marco del avance que se ha obtenido en el desarrollo de la economía del país y el incremento de las relaciones económicas y comerciales, la Cámara de Comercio de la República de Cuba y el Centro de Gobierno Electrónico adscrito a la Facultad 3, de la Universidad de las Ciencias Informáticas (UCI), acordaron la creación de un sistema informático que permita realizar la gestión de la información referente a sus afiliados, denominado Sistema de Informatización del Registro de la Cámara de Comercio (SIRECC, de aquí en lo adelante).

El objetivo principal de este sistema es informatizar los procesos de gestión de la información de las empresas autorizadas a operar en el territorio nacional, así como las personas extranjeras que operan como empresarios en Cuba. Además debe contribuir con la obtención oportuna de información que se requiere para tomar decisiones en los diferentes niveles de la

administración del Estado.

SIRECC es una aplicación de escritorio desarrollada sobre una arquitectura cliente-servidor. A medida que se ha venido empleando y ha sido utilizada por los diferentes usuarios, han ido surgiendo carencias y nuevos aspectos del negocio a incorporar. Razones por las cuales se liberan nuevas versionas del sistema que deben actualizarse en la Cámara de Comercio.

Por lo general, las actualizaciones de software vienen justificadas por varios motivos, los principales radican en proporcionar nuevas funcionalidades o mejoras a las versiones anteriores, así como solucionar problemas de compatibilidad. Aunque es posible actualizar los programas manualmente, lo más sencillo y seguro es hacerlo de manera automática. De forma que el propio sistema busque las actualizaciones, las descargue e instale sin la intervención del usuario.

Existen sistemas desarrollados específicamente con este propósito, los denominados actualizadores automáticos. Son aplicaciones que permiten al usuario, comprobar si hay una nueva versión de su sistema. Si la actualización existe, le brinda la posibilidad de descargar e instalar la nueva versión disponible. De esta manera, se evita realizar el proceso de forma manual (2). Incluso existen programas también, que analizan las aplicaciones más comunes ya instaladas, para detectar las que no están correctamente actualizadas, y facilitar su puesta al día.

En la Cámara de Comercio, las estaciones clientes se encuentran distribuidas en diferentes oficinas y departamentos. El proceso de actualización de SIRECC en las mismas es llevado a cabo de forma manual y solamente por personal encargado de esta tarea. Esto implica el empleo de más tiempo en el proceso de actualización de las estaciones clientes, además de la capacitación al personal encargado de esta tarea. Teniendo en cuenta que en el mejor de los casos el proceso se realice y culmine satisfactoriamente, ya que de lo contrario el tiempo utilizado sería mayor.

Por otro lado también influye negativamente el hecho de que la aplicación no es actualizada simultáneamente en todas las estaciones clientes. De ahí que las diferencias existentes entre las distintas versiones instaladas en cada una, puede traer incoherencias a la hora de ejecutar las tareas desarrolladas por los usuarios finales, provocando más pérdida de tiempo e incompatibilidad entre las versiones, hasta finalizar el proceso de actualización. Estos elementos traen como consecuencia, un mayor esfuerzo por parte de la organización para la

realización de la actualización del software.

La capacidad de identificar qué actualizaciones existen, e implementar selectivamente tales actualizaciones de forma automatizada y oportuna, puede ayudar a mantener a la Cámara de Comercio su productividad. De manera que la informatización de este proceso tiene gran importancia para dicha organización, y más aún en la fase de pruebas de aceptación y piloto, producto de las soluciones a las peticiones de cambio en la que actualmente se encuentra involucrada.

Aunque en ocasiones pudiera considerarse que estas actualizaciones no son del todo necesarias para el buen funcionamiento de SIRECC, debe quedar a consideración de la organización, porque algunos usuarios pueden trabajar sin dificultades, pero otros por la complejidad de su trabajo las requerirán. Informatizar este proceso es de gran importancia porque reduce el tiempo de ejecución de esta tarea. Los usuarios pueden mantener sus equipos al día sin tener que preocuparse de descargar e instalar actualizaciones.

Dada la situación problemática antes planteada se identifica el siguiente **problema a resolver**: ¿Cómo contribuir a la actualización automática de las aplicaciones clientes del Sistema de Informatización del Registro de la Cámara de Comercio de la República de Cuba de forma tal que se disminuya el tiempo de ejecución de la tarea y la diferencia de versiones en las estaciones clientes?

Los procesos de actualizaciones automáticas constituyen el **objeto de estudio**.

Se ha definido como **objetivo general** de la investigación desarrollar un componente de actualizaciones automáticas que permita disminuir el tiempo de ejecución de la tarea y la diferencia de versiones en las estaciones clientes.

El **campo de acción** son los procesos de actualizaciones automáticas del sistema SIRECC.

Para resolver el problema planteado se propone defender la siguiente **idea**: Con el desarrollo de un componente de actualizaciones automáticas para las aplicaciones clientes del Sistema de Informatización del Registro de la Cámara de Comercio de la República de Cuba, se disminuirá el tiempo de ejecución de éstas y la diferencia de versiones del sistema en las aplicaciones clientes.

Para el cumplimiento del objetivo general se trazan los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación.
- Desarrollar la propuesta de solución.
- Validar la solución propuesta.

Para dar cumplimiento a los objetivos específicos se planifican las siguientes **tareas** en el proceso de desarrollo de la investigación:

- Caracterización de las soluciones existentes sobre los mecanismos de actualizaciones automáticas en soluciones de software.
- Definición de la metodología de desarrollo, las herramientas y los patrones a utilizar para el desarrollo de los componentes necesarios.
- Definición del mecanismo necesario para efectuar las actualizaciones de software.
- Describir los requisitos funcionales y no funcionales del componente para actualizaciones de software.
- Diseño de los componentes que formarán la solución.
- Implementación de los componentes que formarán la solución.
- Verificación de la especificación de requisitos de software a partir de la aplicación de métricas.
- Verificación del diseño a partir de la aplicación de métricas Tamaño operacional de clases y Relaciones entre clases.
- Aplicación de pruebas de caja blanca a la solución propuesta.
- Realización de pruebas de integración en Sistema de Informatización del Registro de la Cámara de Comercio de la República de Cuba (SIRECC).

Los métodos empleados en la investigación científica son:

Métodos teóricos:

Analítico – Sintético: posibilita realizar un estudio teórico de la investigación que permita el análisis de documentos y la extracción de elementos importantes acerca del proceso de actualización de aplicaciones. Se utiliza además para la precisión de las características del diseño propuesto, así como para el arribo de las conclusiones de la investigación.

Histórico – Lógico: viabiliza el análisis de las características de las herramientas utilizadas y las tendencias evolutivas actuales de su uso en los proyectos productivos de la Universidad y el mundo, así como la definición de los antecedentes y la vinculación con tecnologías confinadas

para el desarrollo del componente.

Métodos empíricos:

Medición: es usado a través del empleo de métricas de calidad para la verificación de los requisitos, como son: las métricas Estabilidad, Especificidad y Grado de validez, que permiten medir cuán estables, carentes de ambigüedad y correctos están los requisitos identificados. También mediante el empleo de métricas para verificar el diseño como son: el Tamaño operacional de las clases (TOC) y Relaciones entre clases (RC), que permiten medir los niveles de acoplamiento, complejidad de implementación y mantenimiento, cantidad de pruebas y reutilización de las clases del diseño propuesto.

El trabajo está estructurado en tres capítulos y el contenido de los mismos se describe a continuación:

Capítulo 1: Fundamentación Teórica: se realiza el estudio del estado del arte, en Cuba y el mundo sobre las diferentes herramientas existentes y los mecanismos que estas emplean para realizar los procesos de actualizaciones automáticas, así como una descripción de las técnicas, tecnologías, metodologías y herramientas usadas para dar solución al problema.

Capítulo 2: Planificación y diseño: se realiza un análisis de las características del sistema a desarrollar a partir de la problemática planteada. Se describen las principales funcionalidades que brindará el componente de actualizaciones automáticas, así como los requisitos no funcionales. Se obtienen artefactos generados a partir de la aplicación de la metodología de desarrollo utilizada, como son: las historias de usuario, el plan de iteración, el plan de entrega y las Tarjetas de Clase-Responsabilidad-Colaboración (CRC).

Capítulo 3: Implementación y Pruebas: Se describen las principales clases y subsistemas definidos en la implementación. Se verifican los requisitos con la aplicación de las métricas para la Estabilidad, Especificación y Grado de validación de los requisitos. Se realiza una verificación del diseño propuesto con la aplicación de las métricas Tamaño operacional de las clases y Relaciones entre clases. Para terminar la validación del resultado de la investigación, se valida la implementación con la aplicación de las pruebas de caja blanca y pruebas funcionales.

CAPÍTULO 1. Fundamentación Teórica

1. Introducción

En este capítulo se abordarán las principales definiciones que se deben tener en cuenta para desarrollar un componente que realice los procesos relacionados con las actualizaciones automáticas para el Sistema de Informatización del Registro de la Cámara de Comercio. Se reflejará una síntesis de las principales características sobre algunas de las soluciones existentes que presentan similitud con la solución propuesta. Además se realiza un estudio sobre las tecnologías, metodologías y herramientas, con el objetivo de realizar una selección para el desarrollo de la solución propuesta.

1.1. Proceso de actualizaciones automáticas

El desarrollo de sistemas informáticos implica un proceso continuo de mejora, verificaciones y ajustes posteriores. Muchas de las empresas periódicamente ponen a disposición de sus clientes versiones mejoradas de sus programas, formando parte del soporte y mantenimiento del mismo. Algunas de las estrategias empleadas por las empresas en el mercado mundial para maximizar sus ganancias, no se encuentran basadas en la compra del producto, si no en este concepto de pago por el soporte que se brinda. Para esto han creado mecanismos de actualización de los sistemas a través de programas encargados de realizar dichas tareas.

Si bien existen varios conceptos relacionados con el término actualización, según el Diccionario de la Real Academia Española, este vocablo se refiere a actualizar (verbo transitivo): Que se encuentra asociado a hacer actual algo, darle actualidad. Poner al día. Poner en acto (37). Como se mencionaba anteriormente esta actualización puede realizarse en varios campos, en los sistemas de software por ejemplo, como consecuencia del incesante avance que se produce en este ámbito. Ya que los cambios o modificaciones a los cuales está sujeto el sistema en algunos lugares, hace que con el tiempo se estén generando nuevas funcionalidades que requerirán de su correspondiente anotación y actualización respecto a otras versiones que van quedando obsoletas. En la literatura consultada esta actualización en términos de software también es conocida como parche, y se tratan indistintamente.

Si bien los parches suelen ser desarrollados por programadores ajenos a los autores iniciales del proyecto, esto no siempre es así. Un parche puede ser aplicado a varios tipos de archivo ya

sea un binario ejecutable como al código fuente de cualquier tipo de programa, incluso, un sistema operativo.

Algunas de las ventajas de estos parches o actualizaciones son las siguientes:

- Hacen correcciones a errores o fallas encontrados.
- Puede mejorar el desempeño de la aplicación y herramientas adicionales.
- Protegen al equipo de potenciales ataques como virus y/o vulnerabilidad de privacidad.

Por otro lado algunas de las desventajas de las actualizaciones de software son:

- Al actualizar un software puede ocasionar consecuencias no intencionales a otros programas.
- Las actualizaciones de algunas aplicaciones pueden tener un costo en cuanto a lo largo, tedioso y complicado del proceso.
- Algunas actualizaciones son pesadas y disminuir el rendimiento de las estaciones clientes.

Como se describe anteriormente las empresas han creado mecanismos basados en herramientas para actualizar sus sistemas, denominados actualizadores. Los cuales ofrecen una localización para descargar las actualizaciones de los componentes del sistema, las cuales pueden incluir, servicios, arreglos de la seguridad, y mejoras brindadas por los usuarios. Proporcionando automáticamente actualizaciones del producto cuando están disponibles. A continuación se muestra un estudio de algunos de ellos reflejando sus características y restricciones principales.

1.2. Estudio de las soluciones existentes

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, ocasiona una rápida obsolescencia del software (3). La mayoría del software desarrollado no tiende a actualizarse por sí solo, por lo que necesita de componentes o herramientas que se encarguen de realizar este proceso donde se reemplace la versión obsoleta para garantizar su mantenimiento y soporte. Existen diferentes soluciones para actualizar aplicaciones entre las cuáles se analizarán algunas como:

1. Symantec LiveUpdateClient

2. Appupdater
3. ClickOnce
4. Actualizador Sistema Gerencia Hospitalaria
5. Actualizador del Sistema de Gestión Penitenciaria Venezolano
6. SVN
7. CVS

1.2.1. Symantec LiveUpdateClient

Es un actualizador de la empresa Symantec, diseñado para mantener actualizados los diferentes productos que son comercializados por dicha empresa. Esta aplicación crea una lista de los productos Symantec que estén instalados, además de otros datos referentes al cliente que necesita actualizaciones; una vez completada hace una petición al servidor Symantec LiveUpdateServer desde donde descarga los paquetes para completar la actualización (4). Este soporta una conexión directa desde el servidor de la red local, sin necesidad de conectar individualmente cada ordenador, por lo que se ahorra tiempo y ancho de banda. La descarga de este programa es gratuita para cualquier usuario interesado. No está disponible para múltiples plataformas y es un producto privativo, por lo que no es una solución viable para la investigación.

1.2.2. Appupdater

Appupdater es otra de las herramientas de instalación y actualización de programas más utilizada en el mundo que se maneja en línea de comandos. No hay diferencia con los comandos apt-get o yum de Linux, que son la vía tradicional de instalación de aplicaciones en este sistema operativo. Appupdater actualiza periódicamente la base de datos con información sobre cientos de aplicaciones para que el usuario la consulte e instale. Además de ser multiplataforma, incluye soporte para dispositivos USB y se pueden sugerir nuevas aplicaciones que se incluirán en la lista de aplicaciones soportadas. Crear un repositorio personalizado si se desea es otra de las posibilidades que brinda Appupdater.

Aparentemente Appupdater es la solución perfecta para actualizar todos los desarrollos independientes una vez que se cumplan con las reglas de creación de repositorio. Entre las principales características de Appupdater destacan algunos aspectos que hacen desistir de la idea de utilizarlo como por ejemplo (5):

- Appupdater descarga una lista de todas las aplicaciones, lo que instala no es enviado ni mostrado, por lo que no se tiene conocimiento de los ficheros o registros actualizados.

- Durante la instalación del programa se descarga directamente desde el sitio web, no es posible hacer un seguimiento de lo que se está instalando y si falla la conexión ocurre un error.

- Presenta problemas de compatibilidad con versiones en español de programas en Windows u otras versiones.

1.2.3. ClickOnce

Es una tecnología de despliegue de aplicaciones (principalmente Clientes Inteligentes), basada en el concepto de publicación en un servidor Web, FTP o de archivos compartidos, para permitir la descarga, instalación y actualización automática de la aplicación. El núcleo de esta tecnología de despliegue está basado en dos archivos XML: un archivo de declaración de aplicación y un archivo de declaración de despliegue.

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

Application file: contiene un resumen de la información de despliegue en sí misma. Este incluye información de versión, donde encontrar la declaración de la información, y el número de opciones respecto al comportamiento en la clase de despliegue y actualización que se realizará. (6)

Las actualizaciones en ClickOnce aseguran que se realicen de forma completa y satisfactoria, y si esto no es posible, entonces no se realiza la actualización. Los archivos de declaración de aplicación y declaración de instalación que genera ClickOnce están firmados digitalmente, esto proporciona la seguridad de que la aplicación y sus actualizaciones provengan de una fuente segura y fiable. Con ClickOnce se tiene la opción de volver a la versión anterior de la aplicación (rollback version, por su término en inglés) desde el servidor (reemplazando la declaración de instalación) o desde el cliente (en añadir o remover programas). ClickOnce optimiza el proceso de actualización al descargar solo los ficheros que han cambiado, para esto, asocia cada

fichero con una clave única para saber si ha cambiado y también para verificar si un archivo ha sido manipulado (file tampering, por su término en inglés) (7).

De los aspectos de ClickOnce que hacen desistir de su selección se puede mencionar que al publicar una actualización para la aplicación, ésta reemplaza la versión anterior. Por lo tanto los archivos XML que tenían los usuarios antes de la actualización no son tomados en cuenta y prácticamente el usuario pierde la configuración que tenía, y debe volver a configurar. Además de esto se debe añadir que es privativo.

1.2.4. Actualizador Sistema de Información para la Gerencia Hospitalaria

El actualizador Sistema de Información para la Gerencia Hospitalaria (SIGHO), desarrollado en México, ayuda a mantener al día las versiones del sistema de manera más organizada apoyando al personal técnico encargado. Para ejecutar el actualizador del SIGHO es importante cubrir los siguientes pre-requisitos para el correcto funcionamiento de la aplicación.

Es necesario que en el equipo donde se ejecutará la aplicación esté previamente instalado el SIGHO, ya que el actualizador no podrá ejecutarse en el servidor de otra manera. El personal encargado no podrá realizar ninguna operación, sin antes tener una cuenta de usuario del sistema SIGHO.

Con la utilización del actualizador automático SIGHO se evitan cargas de trabajo ya que este reemplaza ejecutables automáticamente, registra y des-registra controles automáticamente, compara versiones de controles y ejecutables para que se utilicen las más actuales y posee un mecanismo de autenticación que evita que personal ajeno realice modificaciones que puedan poner en riesgo el funcionamiento del sistema.

Algunos de los aspectos de SIGHO que hacen desistir de su selección son, que el sistema cuenta con la seguridad de que todos los equipos cuenten con las versiones más actuales, no es posible actualizar varios clientes a la vez y el paquete de actualización debe ser copiado manualmente en el cliente donde se encuentra instalado el SIGHO. (8)

1.2.5. Actualizador del Sistema de Gestión Penitenciaria Venezolano

Esta aplicación es la encargada de escuchar los eventos que provienen del Proveedor de Actualización del Sistema de Gestión Penitenciaria Venezolano y realizar actividades en el cliente que garanticen que la actualización enviada sea instalada y activada correctamente por

lo que se encuentra dentro de los Productos Actualizadores del Proveedor. (9) Cuenta con un visualizador de Información que muestra información sobre el estado de la actualización. Realiza operaciones de restablecimiento de la versión anterior en caso de que falle el proceso de actualización. Es una aplicación multiplataforma.

Sus principales desventajas están dadas por solo facilitar la actualización del Sistema de Gestión Penitenciaria Venezolano y porque necesita detener e iniciar el servidor de aplicaciones para que los nuevos cambios sean puestos en práctica.

1.2.6. SVN

Subversion (SVN por sus siglas en inglés) es un sistema de control de versiones libre y de código fuente abierto, que maneja ficheros y directorios a través del tiempo. Posee un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios.

Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápido sin un único conducto por el cual deban pasar todas las modificaciones. Aunque el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad de este se afecte por la pérdida de ese conducto único, si se ha hecho un cambio incorrecto a los datos, simplemente se deshace ese cambio.

Algunos sistemas de control de versiones son también sistemas de administración de configuración de software. Estos sistemas son diseñados específicamente para la administración de árboles de código fuente, y tienen muchas características que son específicas del desarrollo de software, tales como el entendimiento nativo de lenguajes de programación, o el suministro de herramientas para la construcción de software. Sin embargo, Subversion no es uno de estos; sino un sistema general que puede ser usado para administrar cualquier conjunto de ficheros. Esos ficheros pueden ser código fuente o cualquier otra cosa, desde una lista de la compra de comestibles hasta combinaciones de video digital y más allá (10).

1.2.7. CVS

Concurrent Versions System o como es también conocido CVS, es una aplicación desarrollada bajo la arquitectura cliente-servidor que implementa un control de versiones donde mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un programa permitiendo la colaboración de desarrolladores que estén situados a gran distancia. CVS se ha hecho popular en el mundo de software libre y sus desarrolladores difunden el sistema bajo licencia General Public Licence o GPL.

CVS funciona guardando las versiones del proyecto y su historial en un servidor. Los clientes se conectan al servidor para guardar una copia completa del proyecto y luego eventualmente puedan ir subiendo los cambios.

Dentro de las limitaciones que presenta se puede mencionar que los archivos en el repositorio sobre la plataforma CVS no pueden ser renombrados. Estos deben ser primero agregados con otro nombre y luego eliminados. Además presenta soporte limitado para los archivos Unicode y no trabaja muy bien con ficheros muy grandes o ficheros binarios. (33)

Las aplicaciones que fueron analizadas anteriormente no constituyen una solución factible para implementar el proceso de actualización de SIRECC, debido a que son aplicaciones que actualizan productos de los proveedores, con excepción de SVN y CVS. Esto trae como consecuencia que solo facilite el proceso de actualización a los sistemas para los cuales han sido creados.

Es importante resaltar que todas las soluciones analizadas se tomaron sus aspectos positivos, que servirán de guía para el desarrollo del componente, como es el caso de SVN que es una potente herramienta para el control de versiones, que aparentemente podría ser la solución al problema a resolver, pero este tiene la limitante de que si durante la actualización se produce algún error, como un fallo de conexión, la aplicación quedara inutilizable. Esto se debe a que SVN no realiza una copia de respaldo de los ficheros de la versión anterior de la aplicación, lo que trae como consecuencia que hasta que no exista conexión con el servidor de actualizaciones, no se podrá utilizar nuevamente la aplicación. Otra de las limitantes de SVN es que no realiza un bloqueo sobre la aplicación mientras la actualiza, lo que puede provocar errores a la hora de remplazar los ficheros locales con los descargados del servidor, ya que estos puedan estar en uso. Debido a estas deficiencias, es de vital importancia, para asegurar la correcta actualización de las aplicaciones clientes de SIRECC y ganar así en eficiencia

durante dicho proceso, la implementación del componente de actualizaciones automáticas propuesto como solución de presente investigación.

1.3. Lenguajes

El lenguaje no es más que el estilo y modo de hablar y escribir de cada persona en particular. En la informática un lenguaje está definido como un conjunto de signos y reglas que permite la comunicación con un ordenador (14). Para el desarrollo del componente propuesto de la investigación, fue necesaria la utilización de varios lenguajes de programación, como: el lenguaje de marcado extensible (XML, por sus siglas en inglés) y el lenguaje Java, de los cuales a continuación de describirán sus principales características.

1.3.1. Lenguaje de etiquetado extensible XML

Extensible Markup Language (XML) son las siglas del Lenguaje de Etiquetado Extensible. Surgió como un lenguaje de marcado para sustituir a HyperText Markup Language (HTML). Ambos lenguajes son herederos de Standard Generalized Markup Language (SGML), el lenguaje de marcas estándar para la presentación, la descripción formal y de contenido de los documentos. El desarrollo de XML comenzó en 1996 y desde entonces ha tenido un desarrollo exponencial. En realidad, XML surge del campo empresarial, ya que HTML era un lenguaje poco potente para soportar de un modo eficaz y masivo, la realización de negocios virtuales. Intenta mejorar a HTML y toma como punto de partida el viejo lenguaje SGML, pero lo simplifica para poder trabajar en la Web, por eso se creó XML. (12)

Algunos de los usos del XML son:

- Aplicado a los sitios web: permite separar contenido, presentación y que los mismos datos se puedan mostrar de varias formas sin demasiado esfuerzo.

- Para la comunicación entre aplicaciones: representación de los datos muy simple, fácil de transmitir por la red y estándar. En los últimos tiempos este uso se está haciendo muy popular con el surgimiento de los Servicios web.

- Para la configuración de programas: representación de los datos simple y estándar, en contraposición con los crípticos formatos propietarios.

- Utilizado en los Servicios de Mensajería de Java (JMS, por sus siglas en inglés). El

contenido XML de los mensajes provee una lengua franca, para que todas las partes puedan comprender sin importar qué lenguaje utilicen ni la plataforma sobre la que ejecuten.

Durante el desarrollo de la investigación, XML fue utilizado para la confección de dos ficheros. Uno para guardar la configuración de la aplicación cliente y del componente de actualizaciones, así como los datos referentes al servidor de actualizaciones. El otro para registrar todos los ficheros y directorios que integran la actualización.

1.3.2. Lenguajes de programación

Desde el surgimiento de los lenguajes de programación hasta la actualidad, han surgido una gran variedad con sus características específicas. Cada uno tiene ventajas y desventajas que lo hacen más o menos potente ante una necesidad de software determinada, por lo que no se puede afirmar que exista un lenguaje que cubra todas las necesidades requeridas por un software. De ahí la importancia a la hora de seleccionar el o los lenguajes de programación para un proyecto de software. Se debe realizar un análisis de estos a partir de las necesidades del software. Son muchos los lenguajes que pudieran satisfacer las necesidades del proyecto, pero se decidió solo realizar un análisis de algunos de los utilizados en la UCI. Entre estos se pueden mencionar:

- C#.
- C++.
- Java.

Java es un lenguaje de programación con el que se puede realizar un gran número de programas, y es una plataforma informática creada por Sun Microsystems en 1995. (15)

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente al sistema operativo donde se ejecute. Eso quiere decir, que de hacer un programa en Java, este podrá funcionar en cualquier ordenador del mercado. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple. Esto es posible porque se ha creado una Máquina Virtual de Java (JVM, por sus siglas en inglés) para cada sistema operativo, que hace de intermediario entre el sistema operativo y el programa de Java,

y posibilita que este último se ejecute perfectamente.

Java es uno de los lenguajes más usados para el desarrollo de software. Este ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de lenguajes, como los punteros de C++. Posee una arquitectura neutral. La compilación del software desarrollado en Java es independiente de la arquitectura del ordenador donde se ejecute. Para Java lo único que es dependiente del sistema es la JVM y las librerías fundamentales que sean usadas. La neutralidad de la arquitectura mide en gran medida el grado de portabilidad de este lenguaje. Dentro de las características más importantes de este se puede mencionar:

- Orientado a objetos: como la mayoría de los lenguajes, este cumple con el paradigma de la programación orientada a objetos.

- Seguro: se prescinde la utilización de los punteros con el objetivo de eliminar el uso indebido de recursos en memoria. Además la máquina virtual se encarga de verificar que el software no posea fragmentos de código ilegal (código que falsea punteros, viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto) antes de ejecutarlo.

- Interpretado y compilado a la vez: Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina (bytecodes), semejantes a las instrucciones de ensamblador. Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se haya instalado la JMV.

Además de Java existen otros lenguajes de programación como los mencionados anteriormente, que fueron descartados debido a las dificultades que traen consigo su utilización. Por una parte, C++, es muy eficiente debido a su dualidad como lenguaje de alto y bajo nivel, pero tiene muchos problemas que pueden provocar molestias para los programadores. Una de las más notables es que el uso de la memoria queda completamente en manos del programador. El manejo de punteros y memoria permite un mejor control de la memoria y una buena administración de recursos de computadora, pero también puede llevar a colapsar la aplicación ya que todo su manejo queda en manos del programador.

Por otra parte, C# cuya sintaxis y estructuración es muy parecida a la de C++ o Java, es un

lenguaje seguro ya que incluye en todo el código numerosas restricciones para asegurar su seguridad y no permite el uso de punteros, aunque se puede violar estas restricciones mediante el modificador unsafe. Permite la gestión automática de memoria ya que dispone de un recolector de basura, por lo que no es necesario incluir instrucciones de destrucción de objetos. Pero el hecho de ser un lenguaje nativo de la plataforma Microsoft .NET, y tener sus herramientas privativas, costosas y sobre el sistema operativo Windows, es suficiente para darse cuenta de las limitantes para Cuba y por ende para la UCI.

Teniendo en cuenta las características de todos estos posibles lenguajes y basándose en los factores que anteriormente se mencionan se ha decidido usar Java como el lenguaje de programación para desarrollar el componente.

1.4. Metodologías de desarrollo

Una metodología de desarrollo de software es un conjunto de pasos, procedimientos, técnicas y ayudas a la documentación que deben seguirse para desarrollar software, con el fin de que los proyectos lleguen a ser exitosos desde los puntos de vista de objetivos de negocio, costos, funcionalidad, sencillez y capacidad de soporte. Una metodología define quién debe hacer qué, cuándo y cómo debe hacerlo en un proceso de desarrollo de software. Es una guía que indica qué hacer y cómo actuar cuando se quiere obtener algún tipo de investigación.

Lamentablemente no existe una metodología universal para el desarrollo de software, sin embargo las características propias de cada equipo de desarrollo de software así como las necesidades que expongan los mismos ameritan una configuración personalizada del proceso de desarrollo de software. (22)

Existen dos grandes grupos, con marcadas diferencias, en los cuáles se dividen las metodologías: las tradicionales y los procesos ágiles.

Entre las metodologías más usadas se encuentran El Proceso Unificado de Desarrollo o como se conoce por sus siglas en inglés Rational Unified Process (RUP) y la Programación Extrema o como se conoce por sus siglas en ingles Extreme Programming (XP). A continuación se estará viendo una breve descripción de ambas metodologías de desarrollo de software.

1.4.1. El Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado de Desarrollo es una metodología de desarrollo de software que está

basada en componentes e interfaces bien definidas, y junto con el Lenguaje Unificado de Modelado (UML), utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es un proceso que puede especializarse para una gran variedad de sistemas de software, en diferentes áreas de aplicación, tipos de organizaciones, niveles de aptitud y tamaños de proyecto.

RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de varias metodologías utilizadas por los clientes. La versión que se ha estandarizado vio la luz en 1998, y se conoció en sus inicios como Rational Unified Process, de ahí las siglas con las que se identifica a este proceso de desarrollo. (17)

Como RUP es un proceso, en su modelación define como sus principales elementos:

- Trabajadores (quién): define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.

- Actividades (cómo): es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

- Artefactos (qué): productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

- Flujo de actividades (cuándo): secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

En RUP se han agrupado las actividades en grupos lógicos definiéndose nueve flujos de trabajo principales. Los seis primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. Ver Figura 1.

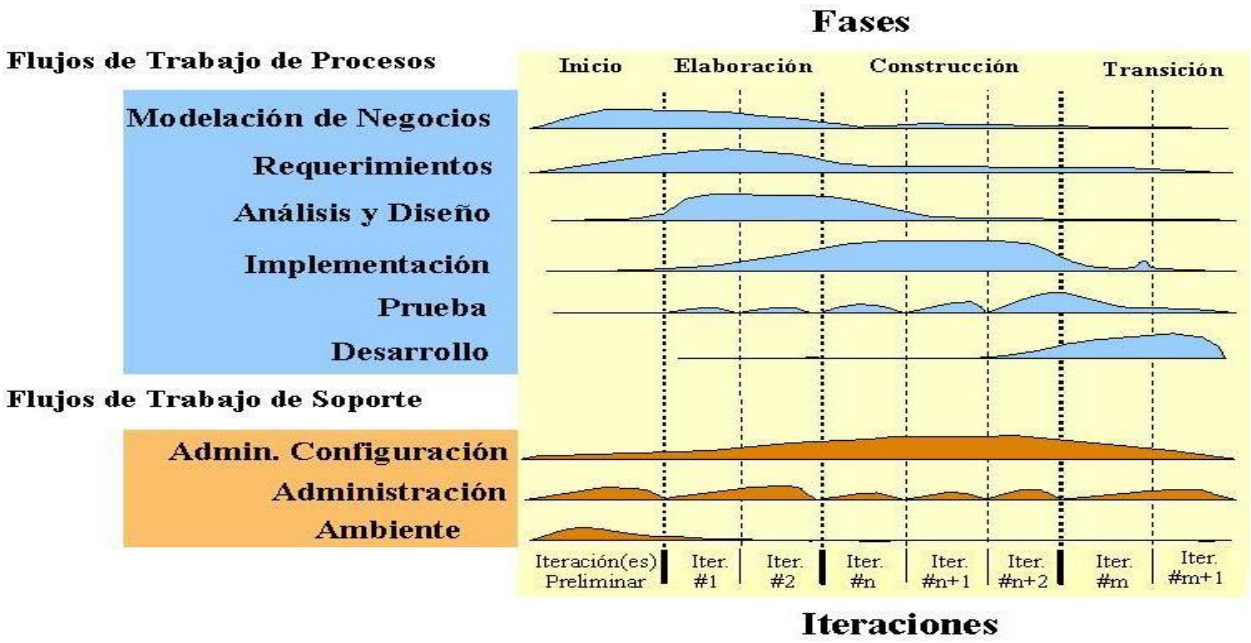

Figura 1 RUP en dos dimensiones.

Entre las características principales de Proceso Unificado de Desarrollo se pueden mencionar:

- Es dirigido por casos de uso: éstos reflejan las necesidades futuras de los usuarios, además capta cuando se modela el negocio y se representa a través de los requerimientos.

- Centrado en la arquitectura: ésta muestra la visión común del sistema completo con la que el equipo del proyecto y los usuarios deben estar de acuerdo. Describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

- Iterativo e incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla algunos más que otros. (18)

La metodología Proceso Unificado de Desarrollo es identificada como perteneciente a los procesos de desarrollo del tipo pesados. Se le llama de esta forma por la extensión en tiempo de desarrollo, así como el gran número de especialistas necesarios para la confección del software. Los negocios de gran magnitud encajan perfectamente en esta metodología puesto que son estos los que involucran a gran cantidad de especialistas de la materia que trate, no

siendo estos los rasgos distintivos del sistema integrado que se propone en la investigación para el proyecto productivo SIRECC. Este no cuenta con un cronograma amplio para su desarrollo así como requiere de un gran número de especialistas en el tema para su desempeño, por lo que esta metodología no se adecua a las necesidades que imponen las características propias del sistema que se propone en la investigación.

1.4.2. Metodología ágil XP

Dentro de las metodologías ágiles existentes, se destaca la metodología de Programación Extrema o como se denomina en inglés Extreme Programming (XP). Esta se centra en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software y promueve el trabajo en equipo. Se define como adecuada para proyectos pequeños. Está basada en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y capacidad para enfrentar los cambios. (19)

XP propone un desarrollo en pareja. Esta estrategia o estilo de programación tiene muchas ventajas: muchos errores son detectados conforme son introducidos en el código; por lo que la cantidad de errores del sistema informático final suelen ser menos, los diseños son mejores y el tamaño del código menor, los problemas de la implementación se resuelven en menos tiempo, se posibilita la transferencia de conocimientos entre los miembros del equipo; pues cada uno entiende las diferentes partes del sistema y por último los programadores o desarrolladores disfrutan más su trabajo. (25)

Esta metodología está basada en:

- Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de manera tal que se pueda prevenir posibles errores que en el futuro pudieran ocurrir.

- Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

- Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro realiza la acción que el otro no está haciendo en ese momento.

La metodología XP propone cuatro actividades básicas para el desarrollo de un buen software las cuales se describen a continuación:

- Codificar: es la única actividad de la que no se puede prescindir porque se necesita codificar y plasmar nuestras ideas a través del código para expresar la interpretación del problema, así se puede utilizar el código para comunicarnos.

- Hacer pruebas: las pruebas dan la oportunidad de saber si lo que se implementó es lo que en realidad se pensaba que se había implementado. Las pruebas indican que el trabajo funciona, cuando no se pueda pensar en ninguna prueba que pudiese originar un fallo en el sistema entonces hemos acabado por completo.

- Escuchar: se tiene que escuchar a los clientes cuales son los problemas de su negocio, se debe tener una escucha activa explicando lo que es fácil y difícil de obtener, y la realimentación entre ambos ayudan a todos entender los problemas.

- Diseñar: el diseño crea una estructura que organiza la lógica del sistema, permite que el sistema crezca con cambios en un solo lugar. Deben de ser sencillos, si alguna parte del sistema es de desarrollo complejo, divídela en varias. Si hay fallos en el diseño o malos diseños deben ser corregidos cuanto antes. (20)

El ciclo de desarrollo de XP consta de seis fases:

- Exploración: en esta fase el cliente define el valor de negocio a implementar. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

- Planificación de la entrega: en esta fase, el programador estima el esfuerzo necesario para su implementación.

- Iteraciones: esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Al final de la última

iteración el sistema estará listo para entrar en producción.

- Producción: la fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación.

- Mantenimiento: mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en desarrollo.

- Cierre del proyecto: es cuando el cliente no tiene más valores del negocio a ser incluidos en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. (21)

Los artefactos generados por la metodología XP son:

- Historias de usuario

- Tarjetas CRC

- Tareas de ingeniería.

La metodología seleccionada es XP debido a su adecuación para el desarrollo del componente propuesto como solución en la investigación, pues dicho sistema debe ser implementado atendiendo a un cronograma de desarrollo ajustado al menor tiempo posible debido a la necesidad de garantizar en la Cámara de Comercio de la República de Cuba las actualizaciones automáticas de sus estaciones clientes. Además XP se basa en la retroalimentación constante entre el cliente y el equipo de desarrollo y propone un desarrollo en pareja, permitiendo que este proceso se realice de manera más rápida y eficiente.

1.5. Protocolos de transferencia

Los protocolos de transferencia de datos son formatos estandarizados para transmitir datos entre dispositivos interconectados en la red. El tipo de protocolo utilizado puede determinar variables como el método de comprobación de errores, el método de compresión de datos y la indicación de fin de archivo. Si todas las redes estuvieran organizadas de la misma forma y todo el software y equipos de las redes se comportaran de manera similar, sólo sería necesario un protocolo para todas las transmisiones de datos. Sin embargo, en la actualidad existen millones de redes distintas con una amplia gama de combinaciones de hardware y software (36). Esto mismo sucede con la red que interconecta las estaciones clientes de la Cámara de Comercio de la República de Cuba, por lo que es necesario brindar soporte para la trasferencia de datos mediante el uso de diferentes protocolos existentes. A continuación son descritos los principales protocolos utilizados como parte de la implementación del componente propuesto de la solución.

1.5.1. Protocolo de transferencia de ficheros FTP

El protocolo de transferencia de ficheros o File Transfer Protocol (FTP por sus siglas en inglés) es uno de los más antiguos dentro de Internet. Es un servicio que los usuarios utilizan frecuentemente (descarga de drivers, música, documentos, etc.), pero esto es solo una parte del servicio, ya que también es posible, implementar en nuestra máquina, un servidor FTP para que otros usuarios se puedan conectar a nuestra computadora y recoger/dejar información en una zona concreta. Existen dos tipos de transferencias:

- Descarga: Consiste en traer un archivo al ordenador desde un servidor remoto. También se dice "bajar un archivo".
- Carga: Consiste en llevar un archivo desde el ordenador a un servidor. También se dice " subir un archivo".

El servicio FTP, es en conclusión un servicio que se utiliza para transferir información, almacenada en ficheros, de una máquina remota a otra local, o viceversa. Para poder realizar esta operación es necesario conocer la dirección IP (o el "nombre") de la máquina a la que nos queremos conectar para realizar algún tipo de transferencia.

Los servidores FTP controlan el acceso de los usuarios a su sistema de carpetas. Esto quiere decir que, para conectarse a un servidor FTP, necesitamos un usuario y una contraseña. Se

ofrece una alternativa de acceso no autentificado: el usuario anonymous, que no lleva contraseña. Normalmente, este usuario tiene los permisos restringidos.

Existen tres tipos de FTP con utilidades muy diversas:

- FTP Anónimo: Cualquiera puede descargar un archivo sin identificarse.
- FTP Autenticado: Hay que conectarse al servidor utilizando un nombre y una contraseña.
- FTP Embebido: Es el que se realiza desde las páginas Web, a través de navegadores. Es una opción dentro del FTP Anónimo. (13)

1.5.1.1. Protocolo seguro de transferencia de ficheros FTPS

Es una extensión de FTP que agrega soporte para los protocolos criptográficos TLS (Transport Layer Security) y SSL (Secure Sockets Layer). FTPS normalmente es más conocido ya que usa los mismos comandos que FTP. Con el uso de este protocolo, la transferencia de archivos es cifrada, agregando seguridad a la misma. Es más seguro que el protocolo FTP y permite una serie de operaciones sobre archivos, ficheros, o carpetas remotas. (27)

1.5.2. Protocolo de transferencia de hipertexto HTTP

Desde 1990, el protocolo de transferencia de hipertexto o Hypertext Transfer Protocol (HTTP por sus siglas en inglés) es el protocolo más utilizado en Internet. La versión 0.9 solo tenía la finalidad de transferir los datos a través de Internet, en particular páginas web escritas en HTML. Este protocolo permite la transferencia de mensajes con encabezados que describen el contenido de los mensajes mediante la codificación MIME.

Se utiliza en las transferencias de información de páginas en Internet, de tal forma que puedan ser visualizadas en un navegador o explorador; habitualmente comprenderá, entre otros elementos, textos en lenguaje HTML, imágenes, Applets de Java, datos, documentos de diversos tipos, animaciones y elementos multimedia.

HTTP es el protocolo de transferencia de información que forma la base de la colección de información distribuida denominada World Wide Web. El protocolo HTTP no fija exactamente lo que se envía o cómo está programado, solo se refiere al mecanismo empleado para hacer llegar y recibir dicha información entre los servidores y el usuario final. Por tanto, controlará el mecanismo de comunicación entre los servidores. (16)

1.5.2.1. Protocolo seguro de transferencia de hipertexto HTTPS

El protocolo de transferencia segura de hipertexto o Hypertext Transfer Protocol Secure (HTTPS por sus siglas en inglés) es una versión segura del protocolo HTTP que implementa un canal de comunicación seguro y basado en Secure Socket Layers (SSL por sus siglas en inglés) entre el navegador del cliente y el servidor HTTP.

El protocolo HTTPS garantiza que la información que sea transmitida entre la computadora del usuario y el sitio web, será cifrada en su transmisión. Equivalente a una carta enviada por correo certificado, lo que se garantiza será entregada a su receptor. (16)

1.6. Herramientas para el desarrollo.

1.6.1. Librería Apache Commons Net.

La librería Apache Commons Net implementa el lado cliente para muchos de los protocolos básicos de internet. El propósito de esta librería es brindar acceso a los principales protocolos de comunicaciones. Esta librería comenzó con un uso comercial bajo el nombre de NetComponents. Después de 1998 el código fuente fue donado a la Fundación Apache y estuvo disponible entonces bajo Licencia Apache. Desde entonces han sido mucho los programadores que han contribuido al desarrollo de Jakarta Commons Net. Actualmente Jakarta Commons Net es un proyecto independiente llamado Apache Commons. Dentro de los protocolos incluye varios como: (34)

- FTP/FTPS
- FTP over HTTP (experimental)
- NNTP
- SMTP(S)
- POP3(S)
- IMAP(S)
- Telnet
- TFTP
- Finger
- Whois
- rexec/rcmd/rlogin
- Time (rdate) and Daytime

- Echo
- Discard
- NTP/SNTP

Para el desarrollo de la investigación esta librería será utilizada para dar soporte al componente de conexión al servidor de actualizaciones mediante los protocolos de trasferencia de ficheros FTP y su versión más segura el protocolo FTPS.

1.6.2. Entorno de Desarrollo Integrado NetBeans 7.3.

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) no es más que un programa compuesto por un conjunto de herramientas para un programador, este puede ser un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los mismos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación y cabe la posibilidad que un mismo IDE pueda funcionar con varios lenguajes de programación. Existen diferentes IDE, pero para la realización del presente trabajo se utilizará el NetBeans, debido a las características relacionadas a continuación.

NetBeans IDE es un entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. Es de código abierto en constante crecimiento que cuenta con una amplia comunidad de usuarios y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones web, empresariales, de escritorio y aplicaciones móviles utilizando la plataforma Java, pero se puede utilizar para desarrollar en otros lenguajes de programación. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos. El mismo puede extender sus funcionalidades mediante la incorporación de módulos gracias a su arquitectura modular. Es un producto libre y gratuito sin restricciones de uso. (23)

Es un entorno de desarrollo que posee un alto grado de robustez y valor para el usuario final, que se traduce en una considerable reducción en el tiempo de desarrollo, consistencia en la interfaz de usuario, independencia de la plataforma, reusabilidad y confiabilidad (24). Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas.

1.7. Conclusiones parciales del capítulo

En este capítulo se determinó, de acuerdo a las potencialidades y características de cada una de las tecnologías antes descritas, el uso de las siguientes para el desarrollo de la investigación: XP como metodología de desarrollo, Java como lenguaje de programación integrado en la plataforma J2EE, NetBeans como entorno de desarrollo integrado en su versión 7.3 y la librería Apache Commons Net. Además se hace necesario el desarrollo de un componente que realice de manera automática las actualizaciones de las aplicaciones SIRECC debido a que con la informatización de este proceso de actualización se reduce su tiempo de ejecución y se disminuye la diferencia de versiones de los sistemas en las estaciones clientes.

CAPÍTULO 2. Planificación y Diseño

2. Introducción

Teniendo en cuenta el análisis del objeto de estudio se decide implementar un componente de actualizaciones automáticas que gestione las actualizaciones del SIRECC. La puesta en práctica de este componente beneficiará a los usuarios del software y a los especialistas de la Cámara de Comercio. En este capítulo se abordarán las principales características del componente de actualizaciones realizando una profunda descripción del mismo. Son mostrados los principales artefactos generados durante las fases de planeación y diseño de la metodología seleccionada para el desarrollo. Son identificados un conjunto de requisitos funcionales y no funcionales basados en las historias de usuario escritas por los clientes y se establecen los planes de iteración y entrega para cada una de las historias de usuario. Se justifican además la selección de los principales elementos dentro del diseño del componente como son: la arquitectura seleccionada, el patrón arquitectónico y los patrones de diseño, así como la obtención de las tarjetas CRC como único artefacto generado durante dicha fase.

2.1. Funcionamiento del componente

A continuación se brinda una descripción detallada del funcionamiento del componente propuesto como solución de la investigación.

2.1.1. Estados del proceso de actualización

A partir del estudio de la problemática se diseñó el proceso de actualizaciones automáticas, el cual está dividido en una serie de estados. A medida que un estado se completa de forma correcta se avanza al siguiente hasta que se termina con el proceso de actualización. Dividir el proceso en varios estados permite más control sobre este y una mejor respuesta ante errores. Los estados en los que se puede encontrar son los siguientes:

1. **Inicio del proceso de actualización:** es el estado inicial en el que se encontrará el actualizador al comenzar. Se hace un bloqueo sobre la aplicación del cliente para que no pueda ser ejecutada mientras se actualiza.

2. **Realizar copia de respaldo:** se hace una copia de respaldo de los ficheros a actualizar.

3. **Descarga de ficheros:** se descargan los ficheros que componen la actualización desde

el servidor.

4. **Sustitución de ficheros:** se eliminan los ficheros de la versión anterior y se copian los que fueron descargados desde el servidor.

5. **Finalización del proceso:** se eliminan ficheros descargados y el fichero de control de la actualización quitando así el bloqueo sobre la aplicación del cliente. Una vez concluido se finaliza el proceso del componente de actualizaciones.

2.1.2. Inicio del proceso de actualización.

Una vez definidos los estados de actualización se diseña el proceso de actualizaciones automáticas con las siguientes características.

El componente será ejecutado por el sistema operativo. Cuando este se inicia, si es Windows ejecutará un servicio encargado de iniciar el componente de actualizaciones o si el sistema es Linux será un demonio encargado de realizar la tarea.

La ejecución del proceso de actualización es el estado inicial en el que se va a encontrar dicho proceso al iniciar el componente. Este estado será registrado en un fichero llamado “actualización.dat” (de ahora en adelante “fichero de control de la actualización”) con el estado 0, el cual gestiona la información del proceso en cada actualización. En esta etapa se notifica al usuario que se ha iniciado el proceso de actualización, luego se desencriptará y se leerá el fichero de configuración XML de la aplicación para obtener los datos del tipo de servidor de actualizaciones (FTP, FTPS, HTTP, HTTPS), puerto, dirección, versión de la aplicación instalada, usuario y contraseña.

Una vez bloqueada la aplicación y obtenidos los datos del servidor de actualizaciones se chequea si la actualización anterior fue satisfactoria o no. Para esto, se verifica la existencia del fichero de control de la actualización. Si existe significa que el proceso anterior no se completó de manera correcta. Al suceder esto, se chequea de nuevo la existencia de nuevas actualizaciones por lo que el componente se conecta al servidor de actualizaciones y descarga el fichero XML de la actualización disponible. Una vez descargado el fichero XML es analizado y se obtienen los datos de la versión y los ficheros que contiene la actualización. La versión de la actualización se compara con la versión actual del cliente. En caso que la aplicación tenga la misma versión que la actualización se pasa al estado final que da por terminado el proceso.

Si existen, se borrarán los datos del proceso anterior y se comenzará de nuevo a actualizar y si no, se reanudará el proceso de actualización a partir de la última acción registrada en el fichero de control de la actualización. Si la última acción que se realizaba era el proceso de copia de respaldo de archivos se chequeará además si el proceso de descarga de ficheros finalizó de forma correcta debido a que ambos se ejecutan de manera simultánea.

Si el fichero de control de la actualización no existe entonces se creará en el directorio donde se encuentra el componente de actualizaciones con los estados inicializados en 0, se comparan las versiones de la aplicación cliente y de la actualización. Si es menor se procede a actualizar, de lo contrario, se pasa al estado final que da por terminado el proceso.

2.1.3. Copia de respaldo y descarga de los ficheros del servidor

Durante el proceso de actualización es necesario realizar una copia de respaldo de los ficheros antiguos que serán actualizados en el cliente. Esta copia servirá para restaurar la aplicación en caso que ocurra un error durante el proceso de actualización y sea necesario restaurar la versión anterior que se tenía. De igual forma se podrá restaurar la versión anterior de la aplicación si la actualización aún contenía errores de la etapa de desarrollo. Este proceso se corresponde con el estado 2 del proceso de actualizaciones. Paralelo a este proceso de copia de respaldo se realizará la descarga de los ficheros de la actualización desde el servidor para ahorrar en tiempo la duración del proceso de actualización. Este proceso de descargas se corresponde con el estado 3.

El problema principal para realizar estos procesos de forma simultánea radica, que no se puede acceder al mismo tiempo de dos procesos diferentes a un mismo fichero. Además, es necesario llevar el control de ellos para en caso que exista un error se pueda continuar, una vez reiniciado el proceso de actualización, por la acción donde se encontraba. Por tanto, para poder realizar estos dos procesos a la vez es necesario auxiliarse de otro fichero para el control de la descarga con la misma estructura que el que se utiliza para el control de la actualización. A este nuevo fichero lo nombraremos “descarga.dat” (a partir de ahora fichero de control de descarga) y con el estado 3.

Al realizar la copia de respaldo, en el fichero de control de la actualización se registrará el estado en que se encuentra el proceso de actualización (en este instante estaría en el estado 2) y además, a medida que se respalde un fichero se registrará su posición en la lista, pues esta mantendrá siempre el mismo orden cada vez que se lea el fichero XML. De igual forma se

procederá al realizar la descarga de los ficheros desde el servidor, y cada vez que se descargue un fichero del servidor se actualizará su posición de la lista del fichero de control de la descarga. Una vez terminado el proceso de descarga este fichero se eliminará como constancia que este proceso se terminó correctamente y se continuará con el proceso de actualización.

En caso que en una actualización realizada con anterioridad se haya producido un fallo y en el fichero de control de la actualización se encuentre en estado 2, para realizar la recuperación al iniciar el componente de actualización es necesario chequear de nuevo si existe una nueva versión de la aplicación en el servidor. Para esto se compara la versión del fichero XML de actualización que se descargó en el momento que se actualizando y ocurrió el error con el fichero de actualización que se encuentra en el servidor. De existir una nueva versión, se eliminan los datos descargados hasta ese momento y los datos de respaldo existentes del cliente. Una vez realizado esto se vuelve a reiniciar el estado 2 pero con la nueva actualización.

Si no hay una nueva versión de la aplicación en el servidor, se chequea entonces la existencia del fichero de control de la descarga. Si este fichero existe entonces significa que también se interrumpió este proceso y es necesario reiniciarlo. De ambos ficheros de control se carga la posición del fichero que estaba siendo respaldado y que se descargaba, así de manera simultánea se reinician los dos procesos por las posiciones donde estaban registradas en los ficheros de control. Una vez terminado el proceso de la descarga se elimina el fichero de control de descarga como constancia que terminó sin errores. Si no existe el fichero de control de descarga se termina de realizar la copia de respaldo de los ficheros que faltaron y se pasa la siguiente etapa del proceso de actualización.

2.1.4. Actualización de ficheros y finalización del proceso de actualización

Cuando se termina el proceso de respaldo de los ficheros que serán actualizados y se terminan de descargar todos los ficheros que componen la actualización, entonces se comienza a actualizar cada uno de los ficheros del cliente que están contenidos dentro de la actualización. Primero en el fichero de control de la actualización se registra el número de la acción que se va a realizar y a medida que se va actualizando un fichero, se va también actualizando su posición. Si durante el proceso de actualización de los ficheros ocurre algún error, entonces se restaurará la copia de respaldo existente y se eliminarán los archivos descargados desde el servidor. Una vez actualizados todos los ficheros, se eliminan los datos temporales

descargados desde el servidor, el fichero de control de la actualización, los archivos respaldados y se le notifica al usuario que finalizó el proceso de actualización.

Fichero de actualización: Es un fichero XML donde están registrados todos los ficheros que integran la actualización, posee el nombre de “actualización.xml”. A su interior se encuentran los datos de los directorios nuevos que deberán ser creados y de los ficheros que componen la actualización. De estos ficheros se registra el nombre y el directorio donde debe ser ubicado.

```
<Actualizacion version="2.0.5">
    <Directorio dir="lib\"/>
    <Fichero
    NombreArchivo="AbsoluteLayout.jar"
    Hash="6f6a9504a7ccfe9ee2c253345a500a91"
    Directorio="lib\" />
    <Fichero
    NombreArchivo="acc-config.jar"
    Hash="762e573aad2b430640123cb8345e20fc"
    Directorio="lib\" />
</Actualizacion>
```

Figura 2 Fichero de actualización

Fichero de configuración de la aplicación: Es el fichero XML donde se guarda la configuración de la aplicación cliente. Además deben guardarse aquí los datos de configuración del componente de actualizaciones. Los datos que serán registrados en este fichero son los referentes al servidor de actualizaciones. Estos son el tipo de servidor, si requiere autenticación, dirección del servidor, puerto, usuario y contraseña. Este fichero se encontrará encriptado en el directorio base de la aplicación.

2.2. Planificación

La metodología de desarrollo de software XP muestra la planificación como un permanente diálogo entre el usuario final y el equipo de desarrollo. Se define lo que el software tiene que resolver para que genere algún valor, la parte de este que debe hacerse primero, lo que se necesita hacer para saber si el negocio requiere, o no, de la informatización a través de un software, fechas de entrega de las primeras versiones del software, la organización del trabajo, y el equipo de desarrollo. Por último, se detalla dentro de una versión del producto los problemas que deben de ser resueltos con carácter urgente. (25)

En esta fase es donde se obtienes los artefactos conocidos como las historias de usuarios, a

partir de los cuales es posible detectar cuáles serán los requisitos funcionales del componente a desarrollar. Además son identificados también los requisitos no funcionales y se establecen los planes de iteración y entrega.

2.3. Historias de usuario

Las historias de usuario son descritas por los propios clientes, tal y como ven ellos las necesidades del sistema. Por tanto serán descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica, estas conducen el proceso de creación de las pruebas de aceptación las cuales se utilizarán para verificar que las historias de usuario han sido implementadas correctamente, la principal diferencia es el nivel de detalle.

El nivel de detalle de las historias de usuario debe ser el mínimo posible que permita hacerse una ligera idea de cuánto costará implementar el sistema, siendo los desarrolladores los que hacen una estimación de cuánto tiempo, les llevará implementar cada una de estas (25). Además proporcionarán los detalles sobre la estimación del riesgo y cuánto tiempo conllevará su implementación.

Historias de usuario detectadas:

- Ejecución del actualizador.
- Chequeo de nuevas actualizaciones en el servidor.
- Descarga de actualizaciones desde el servidor.
- Copia de respaldo de la aplicación.
- Restaurar la versión anterior de la aplicación.

Historia de Usuario	
Número: 1	**Nombre Historias de usuario:** Ejecución del actualizador**.**
Modificación de Historia de Usuario Número:	
Referencia:	
Programador: Yasiel Dueña Sánchez	**Iteración Asignada:** 1
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados(días):** 4
Riesgo en Desarrollo: Alta **(Alto/Medio/Bajo)**	**Puntos Reales(días):** 7
Descripción: Inicia cuando el usuario se identifica en el sistema operativo. Una vez	

identificado, se inicia un servicio encargado de ejecutar el actualizador.
Observaciones:

Tabla 1 Historia de usuario: Ejecución del actualizador

Historia de Usuario	
Número: 2	**Nombre Historias de usuario:** Chequeo de nuevas actualizaciones en el servidor.
Modificación de Historia de Usuario Número:	
Referencia:	
Programador: Yasiel Dueña Sánchez	**Iteración Asignada:** 2
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados (días):** 4
Riesgo en Desarrollo: Alta **(Alto/Medio/Bajo)**	**Puntos Reales (días):** 7
Descripción: Inicia cuando cuándo el sistema operativo ejecuta el componente de actualizaciones. Este debe de conectarse al servidor de actualizaciones y descargar el fichero update.xml de la actualización existente para ser comparado su versión con la de la aplicación instalada.	
Observaciones:	

Tabla 2 Historia de usuario: Chequeo de nuevas actualizaciones en el servidor.

Historia de Usuario	
Número: 3	**Nombre Historias de usuario:** Descarga de actualizaciones del servidor.
Modificación de Historia de Usuario Número:	
Referencia:	
Programador: Yasiel Dueña Sánchez	**Iteración Asignada:** 3
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados(días):** 4
Riesgo en Desarrollo: Alta **(Alto/Medio/Bajo)**	**Puntos Reales(días):** 7
Descripción: El componente, al comprobar la existencia de una nueva actualización en el servidor, inicia la descarga de los ficheros que componen la misma.	
Observaciones:	

Tabla 3 Historia de usuario: Descarga de actualizaciones del servidor.

Historia de Usuario	
Número: 4	**Nombre Historias de usuario:** Copia de respaldo de la aplicación.
Modificación de Historia de Usuario Número:	
Referencia:	
Programador: Yasiel Dueña Sánchez	**Iteración Asignada:** 4
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados(días):** 4
Riesgo en Desarrollo: Alta	**Puntos Reales(días):** 7

(Alto/Medio/Bajo)	
Descripción: El componente, al iniciar la descarga de ficheros desde el servidor, comienza a realizar paralelamente una copia de respaldo de los ficheros de la aplicación que serán actualizados.	
Observaciones:	

Tabla 4 Historia de usuario: Copia de respaldo de la aplicación.

Historia de Usuario	
Número: 5	**Nombre Historias de usuario:** Restaurar la versión de la aplicación.
Modificación de Historia de Usuario Número:	
Referencia:	
Programador: Yasiel Dueña Sánchez	**Iteración Asignada:** 5
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados(días):** 4
Riesgo en Desarrollo: Alta **(Alto/Medio/Bajo)**	**Puntos Reales(días):** 7
Descripción: El componente, en caso de ocurrir un error durante el proceso de actualización de ficheros, restaura la aplicación a su versión anterior mediante la copia de respaldo almacenada.	
Observaciones:	

Tabla 5 Historia de usuario: Restaurar la versión de la aplicación.

2.4. Requisitos

Una vez realizadas, por el cliente, las distintas historias de usuarios se procede a documentar las necesidades del mismo capturando los requisitos identificados a partir de las historias de usuario. Los requisitos son la base para un desarrollo exitoso así como para una plena conformidad con el entregable final. A continuación se muestra el listado de requisitos definidos para el sistema informático propuesto por la investigación.

2.4.1. Requisitos funcionales

Los requisitos funcionales muestran las características requeridas por el sistema informático propuesto, que expresan una capacidad de acción del mismo, una funcionalidad o comportamiento interno; generalmente expresada en una declaración en forma verbal. (26)

A continuación se listan los requisitos funcionales detectados para el sistema:

- El componente debe permitir al usuario, modificar parámetros de acceso al repositorio.
- El componente debe ser capaz, de forma autónoma, de desencriptar el fichero de configuración de la aplicación.

- El componente debe ser capaz de descargar ficheros de la actualización del repositorio.
- El componente debe ser capaz conectarse a repositorios del tipo FTP, FTPS, HTTP y HTTPS.
- El componente debe ser capaz de notificar el inicio del proceso de actualización.
- El componente debe ser capaz de notificar el fin del proceso de actualización.
- El componente, antes de continuar con un proceso de actualización interrumpido, debe ser capaz de volver a chequear la existencia de actualizaciones.
- El componente debe ser capaz de restaurar a la versión anterior de la aplicación si ocurre un error mientras se actualiza.
- El componente debe ser capaz de revertir la actualización.
- El componente debe ser capaz de continuar por el último estado que estaba ejecutando, si el proceso de actualización fue interrumpido en algún momento.
- El componente debe ser capaz de informar al usuario una vez terminada la actualización.

2.4.2. Requisitos no funcionales

Los requisitos no funcionales definen propiedades del sistema informático que se desea como producto final. Estos requisitos son normalmente a los que debe apuntar la arquitectura y si estos no son cumplidos, el sistema informático propuesto puede no funcionar o el cliente simplemente no aceptar el producto (26). Los requisitos no funcionales identificados para el sistema son:

Usabilidad

RNF1. El sistema está diseñado para ser utilizado por usuarios con pocos conocimientos informáticos.

Confiabilidad

RNF2. Cuando ocurra una excepción, durante el proceso de actualización, se restablecerá la información al estado anterior de forma que no queden inconsistencias en la misma.

Eficiencia

RNF3. El componente debe ser capaz de realizar las actualizaciones en función de disminuir el tiempo de ejecución de la tarea.

Soporte

RNF4. La configuración de las opciones de conexión al servidor de actualizaciones requiere una preparación previa de los usuarios para su correcta explotación.

Seguridad

RNF5. La conexión al servidor de actualizaciones podrá ser realizada de manera segura mediante la utilización de los certificados de seguridad de los protocolos de transferencia HTTPS y FTPS.

Requisitos de Software

RNF6. Instalar en las estaciones de trabajo el software necesario para el correcto funcionamiento del sistema como:

- Máquina virtual de java en su versión 1.6 o superior.

Requisitos de Hardware

RNF7. Proporcionar características mínimas de hardware a las estaciones de trabajo. Las características técnicas mínimas de hardware deben ser las siguientes:

- 1 GB de RAM.
- 30 GB de capacidad en disco duro.
- Adaptador de red Ethernet 100 Mbps.
- Sistema de Energía Ininterrumpida (UPS) 500 Va.

RNF8. Para los servidores se deben proporcionar las características mínimas de hardware siguientes:

- 4 GB de RAM.
- 80GB de capacidad en disco duro para el servidor de actualizaciones.
- Adaptador de red Ethernet 100 Mbps.

- Sistema de Energía Ininterrumpida (UPS) 500 Va.

2.5. Plan de Iteración

Tomando como base cada una de las historias de usuarios y el esfuerzo que se requiere para el desarrollo de estas, se procede a fragmentar el trabajo en distintas iteraciones; obteniendo un trabajo incremental, donde la piedra angular es la comunicación entre el equipo de desarrollo y el cliente. El equipo de desarrollo precede a dividir la confección del sistema informático en 5 iteraciones; las cuales equilibrarán las distintas secuencias de trabajo.

A continuación se describen cada una de las iteraciones propuestas donde la duración total de iteraciones en días se obtiene a partir del esfuerzo en días estimado por el desarrollador para implementar cada historia de usuario:

Iteraciones	**Historias de Usuario**	**Duración total iteraciones (días)**
Iteración 1	Ejecución del actualizador.	7
Iteración 2	Chequeo de nuevas actualizaciones en el servidor.	7
Iteración 3	Descarga de actualizaciones del servidor.	7
Iteración 4	Copia de respaldo de la aplicación.	7
Iteración 5	Restaurar la versión de la aplicación.	7

Tabla 6 Plan de Iteración

2.6. Plan de entrega

Ya elaboradas por el cliente las distintas historias de usuarios y confeccionado por los desarrolladores el plan de iteración se procede a la confección del plan de entrega con la intención que estos obtengan una estimación real en cuanto a nivel de detalle se refiere, fijándose un periodo de tiempo sobre el cual se debe de implementar cada historia de usuario definiéndose su grado de dificultad. El Plan de Entrega posibilita la obtención de una clasificación teniendo en cuenta el riesgo que se corre a la hora de implementar la historia. Estos datos se almacenan en campos que permanecen vacíos en la historia de usuario, el responsable de llenar estos datos es únicamente el programador una vez que haya hecho el análisis requerido de estos.

A continuación se presenta el plan de entrega elaborado por el equipo de desarrollo donde se reflejan las fechas de entregas para las primeras versiones de las historias de usuario:

Artefacto	Hito	Entrega
Ejecución del actualizador.	Final de la primera iteración.	02/04/2013
Chequeo de nuevas actualizaciones en el servidor.	Final de la segunda iteración.	09/04/2013
Descarga de actualizaciones del servidor.	Final de la tercera iteración.	23/04/2013
Copia de respaldo de la aplicación.	Final de la cuarta iteración.	30/04/2013
Restaurar la versión de la aplicación.	Final de la quinta iteración.	7/05/2013

Tabla 7 Plan de entrega.

2.7. Diseño

En la metodología de desarrollo de software XP se aprecia una simplicidad en el diseño, pues esta permite elegir una “metáfora” de sistema, un conjunto de nombres ilustrados de la realidad, al usar tarjetas CRC. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra debe ser removido inmediatamente. (25)

A continuación son descritos los principales elementos que componen al diseño como son: la arquitectura o estilo arquitectónico, el patrón arquitectónico y los patrones de diseño, seleccionados para el desarrollo del componente propuesto como solución de la investigación, así como son obtenidas cada una de las tarjetas CRC.

2.7.1. Arquitectura

La arquitectura de software es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones. (28)

2.7.1.1. Estilo arquitectónico Cliente-Servidor

El estilo arquitectónico o la arquitectura seleccionada para el desarrollo del componente es la arquitectura Cliente-Servidor. Puede ser definida como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma.

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio). En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras. En otras palabras la arquitectura Cliente-Servidor es una extensión de programación modular en la que la base fundamental es separar una gran pieza de software en módulos con el fin de hacer más fácil el desarrollo y mejorar su mantenimiento. Esta arquitectura permite distribuir físicamente los procesos y los datos, en forma más eficiente, lo que en computación distribuida afecta de manera directa el tráfico de la red y lo reduce mucho.

Uno de los aspectos que más ha promovido el uso de sistemas Cliente-Servidor, es la existencia de plataformas de hardware cada vez más baratas. Esta constituye a su vez una de las más palpables ventajas de este esquema, la posibilidad de utilizar máquinas considerablemente más baratas que las requeridas por una solución centralizada, basada en sistemas grandes. Además, se pueden utilizar componentes, tanto de hardware como de software, de varios fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.

El esquema Cliente-Servidor facilita la integración entre sistemas diferentes y comparte información permitiendo, por ejemplo, que las máquinas ya existentes puedan ser utilizadas, pero utilizando interfaces más amigables al usuario. De esta manera, podemos integrar PCs con sistemas medianos y grandes, sin necesidad de que todos tengan que utilizar el mismo sistema operacional (29). En la figura 3, se aprecia un diagrama de despliegue donde se describe la distribución física de los componentes que integran la solución, tanto del lado del cliente y como del servidor.

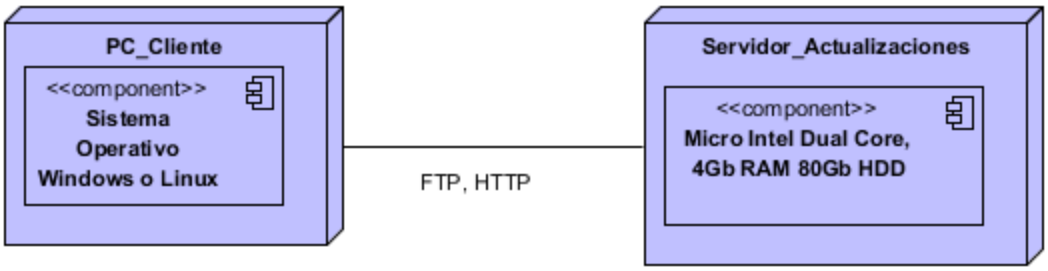

Figura 3 Diagrama de despliegue

2.7.1.2. Estilo arquitectónico 2-capas

El estilo arquitectónico 2-Capas, seleccionado para el desarrollo del componente, tiene objetivo primordial la separación del software en diferentes capas, de manera que a cada nivel se asigna una responsabilidad, permitiendo que el diseño de la arquitectura sea escalable, o sea, que pueda ampliarse con facilidad en caso de que las necesidades aumenten con el desarrollo.

La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se modifica en el nivel requerido sin tener que alterar código de otras capas.

Las capas en las que fue separado el componente objeto son:

- **Datos:** que representa la información con la que trabaja la aplicación, es decir, los ficheros de configuración del componente.
- **Negocio:** que se encarga de procesar los ficheros XML de la actualización y de la configuración, así como crear el hilo para las descargas de ficheros y procesar las operaciones de actualización.

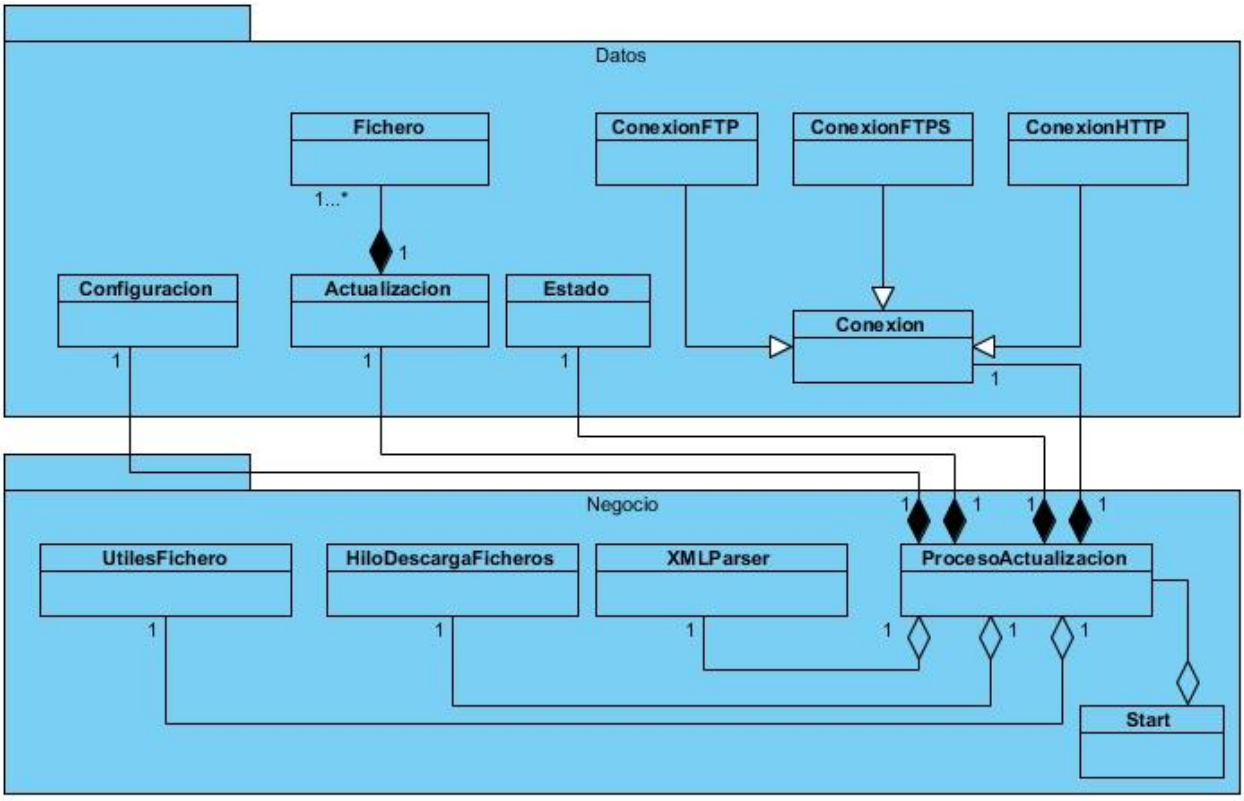

Figura 4 Diagrama de clases por paquetes.

2.7.2. Patrones de diseño

Los patrones de diseño son una guía para obtener soluciones a diferentes problemas presentados en el desarrollo de un software. Los mismos permiten la aplicación de diseños ya probados a diferentes escenarios problemáticos. Para la implementación del componente de actualizaciones automáticas se tuvieron en cuenta los patrones GRASP[1] y GoF[2] los cuales se describen a continuación. (32)

2.7.2.1. Patrones GRASP

Los patrones GRASP, más que patrones propiamente dichos, son una serie de “buenas prácticas” de aplicación recomendable en el diseño de software. Entre ellos, se pueden mencionar los patrones Experto, Creador, Alta cohesión y Controlador, porque guardan directa relación con la creación y asignación de responsabilidades a los objetos. Todo patrón tiene un

[1] Acrónimo de GRASP: Patrones para Asignar Responsabilidades.

[2] Acrónimo de GoF: Gang Of Four.

nombre, plantea un problema y aporta una solución. (32)

Alta cohesión: es una medida que determina cuan relacionadas y adecuadas están las responsabilidades de una clase, de manera que no realice un trabajo colosal; una clase con baja cohesión realiza un trabajo excesivo, haciéndola difícil de comprender, reutilizar y conservar (32). En la figura 4 se aprecia el uso de este patrón de diseño mediante la clase Actualizacion, la cual solo contiene los parámetros y funcionalidades que están estrechamente relacionados con ella, de manera que esta no realice un trabajo excesivo dentro del funcionamiento del componente.

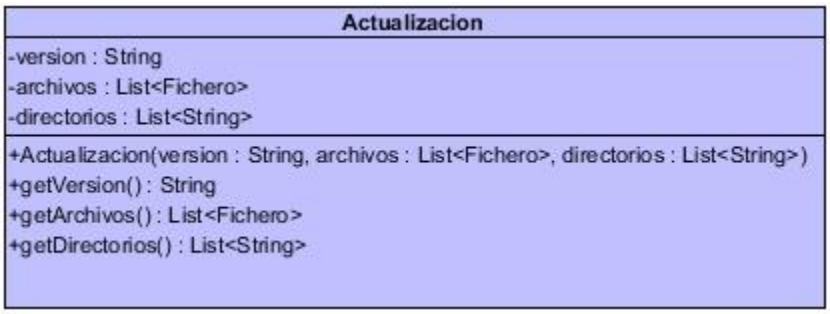

Figura 5 Utilización del patrón Alta Cohesión en la clase Actualizacion

Experto: posibilita una adecuada asignación de responsabilidades facilitando la comprensión del sistema, su mantenimiento y adaptación a los cambios, con reutilización de componentes (32). En la figura 5 se aprecia la utilización de este patrón de diseño mediante la clase UtilesFichero, la cual es la única responsable, dentro del componente, de realizar todas las operaciones sobre los ficheros que componen el proceso de actualización.

Figura 6 Utilización del patrón Experto mediante la clase UtilesFichero

Controlador: una clase controladora sirve intermediaria entre una determinada interfaz y el algoritmo que la implementa, de tal forma que sea la que reciba los datos los envíe a las

distintas clases según el método invocado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, con el objetivo de aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. Este patrón se utiliza porque se hace uso de una clase controladora denominada ProcesoActualizacion (ver Figura 6), la cual centraliza todas la funcionalidades necesarias dentro del proceso de actualización que realiza el componente (32).

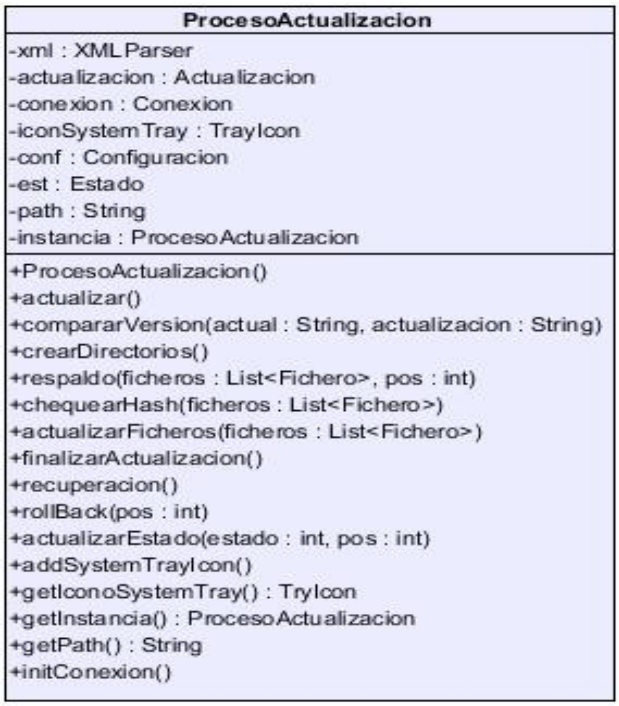

Figura 7 Utilización del patrón Controlador mediante la clase ProcesoActualizacion

2.7.2.2. Patrones GoF

Patrones de Creación

Singleton: garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia (32). Un ejemplo de este patrón se ve reflejado en la figura 7, donde se refleja el código de la clase ProcesoActualizacion dentro de la cual se aprecia la creación de una única instancia de dicha clase y la creación de un método de acceso global a dicha instancia.

```
public class ProcesoActualizacion {

    private static ProcesoActualizacion inscancia = new ProcesoActualizacion();

    private ProcesoActualizacion() {
        this.xml = new XMLParser();
        try {
            this.conf = xml.getDatosConfiguracion();
        } catch (JDOMException | IOException ex) {
            System.exit(0);
        }
        this.est = new Estado();
        this.initConexion();
    }

    public static ProcesoActualizacion getInstancia() {
        return inscancia;
    }
```

Figura 8 Utilización del patrón Singleton mediante la clase ProcesoActualizacion

2.8. Tarjetas CRC (Clase o Cargo, Responsabilidad y Colaboración)

Las tarjetas CRC se definen como un mecanismo eficaz para pensar en el software en un contexto orientado a objetos. Las mismas, identifican, organizan y describen las clases orientadas a objetos que son relevantes para el incremento actual del software, y son el único producto de trabajo de diseño que se generan como parte del proceso XP.

El nombre de la clase se escribe en la parte de arriba de la tarjeta, a modo de título, en una columna a la izquierda se escriben las responsabilidades u objetivos que debe cumplir dicha clase, y a la derecha se escriben las clases que colaboran con cada responsabilidad. (25)

A continuación se muestran las clases de diseño identificadas por el equipo de desarrollo así como las tarjetas CRC correspondientes a algunas de estas. El resto de las tarjetas están reflejadas en los anexos.

Clases:

- Actualizacion
- Configuracion
- Fichero
- ProcesoActualizacion
- Conexión

- ConexionFTP
- ConexionFTPS
- ConexionHTTP
- Estado
- HiloDescargaFicheros
- UtilesFichero
- XMLParser

Tarjeta CRC correspondiente a la clase Actualizacion que por su significación desde el punto de vista del negocio se decide mostrar. La clase tiene la responsabilidad de gestionar la información necesaria para realizar el proceso de actualización y está conformada por tres funcionalidades que no poseen dependencias con otras clases.

Clase: Actualizacion

Responsabilidad	**Clases relacionadas**
Obtener lista de archivos a actualizar.	-
Obtener lista de directorios a crear.	-
Obtener versión de la actualización.	-

Tabla 8 Tarjeta CRC: Actualizacion

Tarjeta CRC correspondiente a la clase Configuracion que por su significación desde el punto de vista del negocio se decide mostrar. La clase tiene la responsabilidad de gestionar la información de la configuración del componente de actualizaciones y está conformada por seis funcionalidades que no poseen dependencias con otras clases.

Clase: Configuracion

Responsabilidad	**Clases relacionadas**
Obtener tipo de servidor.	-
Obtener si requiere autenticación.	-
Obtener el usuario para la autenticación en el servidor de actualizaciones.	-
Obtener la contraseña para la autenticación en el servidor de actualizaciones.	-
Obtener la dirección del servidor de actualizaciones.	-

Obtener el puerto de conexión del servidor de actualizaciones.	-

Tabla 9 Tarjeta CRC: Configuracion.

2.9. Conclusiones parciales del capitulo

En este capítulo se determinó, de acuerdo a las potencialidades y características de cada uno de los elementos correspondientes a las fases de planeación y diseño antes descritos, la utilización de los siguientes para el desarrollo de la investigación: arquitectura Cliente-Servidor como estilo arquitectónico distribuido en capas, estilo arquitectónico 2-Capas estrechamente ligado a la arquitectura seleccionada. Se utilizaron los patrones GoF y GRASP como respuesta a dificultades presentadas dentro del diseño del componente. Además se obtuvieron un conjunto de requisitos funcionales a partir de los artefactos historias de usuario descritas por el cliente, se establecieron los planes de iteración y entrega para cada una de las funcionalidades del componente, y se obtuvo como único artefacto correspondiente al diseño a las tarjetas CRC.

CAPÍTULO 3. Implementación y Pruebas

3. Introducción

En este capítulo serán abordados los principales artefactos generados durante las fases de implementación y pruebas de la metodología seleccionada para el desarrollo del componente, como son: las tareas de ingeniería y los casos de pruebas realizados a cada una de las funcionalidades del sistema. Además se justifica la selección de las métricas para la validación de los requisitos identificados y del diseño seleccionado, evaluando y mostrando los resultados de la aplicación de las mismas. Son aplicadas sobre el código del método principal del componente las pruebas de caja blanca, evaluando y mostrando los resultados de las mismas, y por último son realizadas las pruebas funcionales al componente obtenido para validar los resultados de la investigación.

3.1. Tareas de Ingeniería

Las historias de usuarios están divididas en distintas tareas, las cuales serán implementadas por los desarrolladores, dentro del equipo de desarrollo, aplicando la práctica de la programación en parejas. Cada tarea es detallada especificando la historia de usuario relacionada, programador asignado, tiempo estimado, una breve descripción así como la fase en que se encuentra.

Por su significación a los efectos del negocio, a continuación se detallan aquellas tareas asociadas a cada una de las historias de usuario.

Distribución de tareas por cada historia de usuario.

Historia de Usuario	Tareas
Ejecución del actualizador.	• Crear el servicio de ejecución del componente de actualizaciones.
Chequeo de nuevas actualizaciones en el servidor.	• Establecer la conexión con el servidor de actualizaciones.
Descarga de actualizaciones desde el servidor.	• Garantizar que se descarguen de forma correcta e íntegra los

	ficheros que componen la actualización.
Copia de respaldo de la aplicación.	• Garantizar que se realice de forma correcta la copia de respaldo de los ficheros de la aplicación que serán actualizados.
Restaurar la versión anterior de la aplicación.	• Permitir restaurar la versión anterior de la aplicación en caso de producirse un error durante la actualización.

Tabla 10 Distribución de tareas por cada historia de usuario

3.2. Tareas de ingeniería detalladas

Tarea de Ingeniería	
Número: 1	**Nombre Historia de usuario:** Ejecución del actualizador**.**
Nombre de la tarea: Crear el servicio de ejecución del componente de actualizaciones.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 2
Fecha inicio: 15/4/2013	**Fecha fin:** 17/4/2013
Programador responsable: Yasiel Dueñas Sánchez	
Descripción: Al instalarse la aplicación, debe crearse en dependencia del sistema operativo un nuevo servicio o demonio que se encargue de ejecutar de forma automática el componente de actualización.	

Tabla 11 Descripción de la tarea de ingeniería #1

Tarea de Ingeniería	
Número: 2	**Nombre Historia de usuario:** Chequeo de nuevas actualizaciones en el servidor.
Nombre de la tarea: Establecer la conexión con el servidor de actualizaciones.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 2
Fecha inicio: 25/4/2013	**Fecha fin:** 27/4/2013
Programador responsable: Yasiel Dueñas Sánchez	
Descripción: Se crean mediante el código los elementos necesarios para establecer la conexión con el servidor de actualizaciones.	

Tabla 12 Descripción de la tarea de ingeniería #4

Tarea de Ingeniería	
Número: 3	**Nombre Historia de usuario:** Descarga de actualizaciones desde el servidor.
Nombre de la tarea: Garantizar que se descarguen de forma correcta e íntegra los ficheros que componen la actualización.	

Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1
Fecha inicio: 29/4/2013	**Fecha fin:** 30/4/2013
Programador responsable: Yasiel Dueñas Sánchez	
Descripción: Se crean mediante el código los elementos necesarios para mediante la conexión existente con el servidor de actualizaciones, se descarguen los ficheros que componen la actualización.	

Tabla 13 Descripción de la tarea de ingeniería #5

Tarea de Ingeniería	
Número: 4	**Nombre Historia de usuario:** Copia de respaldo de la aplicación.
Nombre de la tarea: Garantizar que se realice de forma correcta la copia de respaldo de los ficheros de la aplicación que serán actualizados	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 2
Fecha inicio: 2/5/2013	**Fecha fin:** 4/5/2013
Programador responsable: Yasiel Dueñas Sánchez	
Descripción: Se crean mediante el código los elementos necesarios para realizar la copia de respaldo de los ficheros de la aplicación que serán actualizados.	

Tabla 14 Descripción de la tarea de ingeniería #6

Tarea de Ingeniería	
Número: 5	**Nombre Historia de usuario:** Restaurar la versión anterior de la aplicación.
Nombre de la tarea: Permitir restaurar la versión anterior de la aplicación en caso de producirse un error durante la actualización.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 4
Fecha inicio: 4/5/2013	**Fecha fin:** 8/5/2013
Programador responsable: Carlos Ernesto del Junco La Rosa	
Descripción: Se crean mediante el código los elementos necesarios para en caso de presentarse errores durante la actualización de la aplicación, pueda ser restaurada a su versión anterior.	

Tabla 15 Descripción de la tarea de ingeniería #7

3.3. Validación de los resultados

Uno de los pilares de la Programación Extrema es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones dividiendo dichas pruebas en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas funcionales destinadas a evaluar si al final de una iteración

se consiguió la funcionalidad requerida diseñadas por el cliente final. (25)

3.3.1. Métricas

Las métricas son la maduración de una disciplina, que ayudan a la evaluación de los modelos de análisis y de diseño, en donde proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, y contribuyen al diseño de pruebas más efectivas. Es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. Proporcionan una manera sistemática de valorar la calidad basándose en un conjunto de reglas claramente definidas. (22)

3.3.1.1. Métricas para la validación de requisitos

Para llevar a cabo la validación de la calidad de los requisitos se tendrán en cuenta las métricas asociadas a la estabilidad, especificidad y grado de validación de los requisitos, las cuales determinan si el levantamiento de requisitos se efectuó de manera correcta.

3.3.1.1.1. Estabilidad de los requisitos

El objetivo de esta métrica es medir cuan estables son los requisitos para asegurar su adecuación antes de pasar a la próxima disciplina.

Se considera que los requerimientos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación. La estabilidad de los requerimientos se calcula como:

$$ETR = \left[\frac{RT - RM}{RT} \right] * 100$$

Ecuación 1 Estabilidad de los requisitos.

Donde ETR representa la estabilidad de los requisitos a calcular, RT al total de requisitos definidos y RM a la cantidad de requisitos modificados.

ETR = [(11 – 0) / 11] * 100
ETR = [1 – 0] * 100
ETR = 1 * 100
ETR = 100

Esta métrica ofrece valores entre 0 y 100. El mejor valor de ETR es el más cercano a 100 porque mostrará que no se están realizando cambios sobre los requisitos, son estables y, por tanto, es confiable trabajar el análisis y diseño sobre ellos.

3.3.1.1.2. Especificidad de requisitos

El objetivo de esta métrica es cuantificar la especificidad o falta de ambigüedad en la definición de los requerimientos. Para calcular esta métrica deben contarse los requerimientos que tuvieron igual interpretación por los revisores y compararlos con el total de requerimientos definidos. La especificidad de los requerimientos se calcula como:

$$Q_1 = \frac{n_{ui}}{n_r}$$

Ecuación 2 Especificidad de requisitos.

Donde Q1 representa la especificidad de los requisitos a calcular, Nr al total de requisitos definidos y Nui al total de requisitos para los que los revisores tuvieron interpretaciones idénticas.

Q1 = 11 / 11
Q1 = 1

El valor de esta métrica debe estar siempre entre 0 y 1. Mientras más cerca de 1 esté el valor de ER mayor será la consistencia de la interpretación de los revisores para cada requerimiento y menor será la ambigüedad en la especificación de los requerimientos.

3.3.1.1.3. Grado de validación de los requisitos

Los requerimientos deben ser posibles de validar. La validación de los requerimientos se realiza en consenso del equipo de desarrollo al contrastar lo que desea el cliente con la posibilidad real de implementarlo. El grado de validación de los requerimientos mide la corrección en la definición de los requerimientos. Este valor se calcula como:

$$Q_3 = \frac{n_c}{(n_c + n_{nv})}$$

Ecuación 3 Grado de validación de los requisitos.

Donde Q3 representa el grado de validación de los requisitos a calcular, Nc al total de requisitos validados correctamente y Nnv al total de requisitos no validados.

Q3 = 11 / (11 + 0)
Q3 = 11 / 11
Q3 = 1

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo, es el más cercano a 1 e indica un alto nivel de corrección en la definición de los requerimientos.

3.3.1.2. Métricas para la validación del diseño

Las métricas de diseño permiten medir de forma cuantitativa la calidad de los atributos internos del software, permitiendo evaluar la calidad durante el desarrollo del sistema. Se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el proceso de desarrollo de software. Sus objetivos están inclinados a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo. (22)

El grado de calidad y fiabilidad del diseño alcanzado por el sistema se midió a través de métricas basadas en las clases; las cuales miden categorías como tamaño operacional de clase y relaciones entre clases, permitiendo el resultado de ambas métricas validar el diseño propuesto en la investigación.

A continuación se muestra como se aplicaron las métricas Tamaño Operacional de Clase (TOC) y Relación entre Clases al sistema informático.

3.3.1.2.1. Tamaño operacional de las clases (TOC)

El tamaño operacional de una clase está dado por el número de métodos asignados a dicha clase, y evalúa los siguientes atributos de calidad:

Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto de la problemática propuesta. Un aumento del TOC implica un aumento de la responsabilidad asignada a una clase determinada. Si existen valores grandes de TOC significa que una clase tiene demasiada responsabilidad, lo cual reduciría la reutilización de la clase y hará complicada la implementación. De forma contraria sucede si los valores de TOC son pequeños.

Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado. Un aumento del TOC implica un aumento de la complejidad de implementación de una determinada clase.

Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software. Un aumento del TOC implica una disminución del grado de reutilización de una determinada clase. (22)

Las clases que juegan un papel importante en los procesos del sistema informático propuesto en la investigación se encuentran relacionadas en el controlador y en el modelo, siendo aplicada a estas clases las métricas antes descrita.

A continuación se muestran las clases que juegan un papel fundamental en los procesos principales del sistema informático:

Modulo	Clase	Cantidad de procedimientos
Actualizador	Actualizacion	3
Actualizador	Configuracion	6
Actualizador	Fichero	2
Actualizador	ProcesoActualizacion	5
Actualizador	Conexión	5
Actualizador	ConexionFTP	1
Actualizador	ConexionFTPS	1
Actualizador	ConexionHTTP	1
Actualizador	ConexionHTTPS	1
Actualizador	Estado	4
Actualizador	HiloDescargaFicheros	2
Actualizador	UtilesFichero	5
Actualizador	XMLParser	2

Tabla 16 Clases fundamentales

Los umbrales que referencian las afectaciones de los atributos de calidad: responsabilidad, complejidad de implementación y reutilización, los cuales fueron aplicados a las 13 clases del sistema informático, con un promedio de 2.92 procedimientos, quedan especificados en las

siguientes tablas:

Para determinar el grado de afectación de los atributos de calidad que mide la métrica TOC es necesario determinar el tamaño general de cada clase (TGC). El tamaño general de una clase se calcula como:

TGC = NA + TO

Donde NA representa el número de atributos que posee la clase (tanto atributos heredados como atributos privados de la instancia), y TO el total de operaciones que posee la clase (tanto operaciones heredadas como operaciones privadas de la instancia).

Una vez determinado el TGC, se procede a comparar el resultado obtenido con los valores umbrales que aparecen en la Tabla 17 para determinar el TOC de cada clase cuyos resultados son mostrados en la Tabla 21. Por último se calcula el promedio correspondiente a los umbrales, y por consiguiente la afectación de los atributos de calidad establecidos, según lo que se muestra en las Tablas 18, 19 y 20.

Clasificación	Valores de umbrales
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Tabla 17 Clasificación de las clases según el umbral.

Atributo	Categoría	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.

Tabla 18 Rango de valores para medir la afectación del atributo responsabilidad.

Atributo	Categoría	Criterio
Complejidad de implementación	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Tabla 19 Rango de valores para medir la afectación del atributo mantenimiento.

Atributo	Categoría	Criterio
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.

Tabla 20 Rango de valores para medir la afectación del atributo reutilización.

Clase	Cantidad total de Atributos	Cantidad total de Procedimientos	TOC de cada clase
Actualizacion	3	3	Pequeño
Configuracion	7	6	Pequeño
Fichero	2	2	Pequeño
ProcesoActualizacion	6	5	Pequeño
Conexión	4	1	Pequeño
ConexionFTP	6	1	Pequeño
ConexionFTPS	9	1	Pequeño
ConexionHTTP	6	1	Pequeño
ConexionHTTPS	6	1	Pequeño
Estado	2	4	Pequeño
HiloDescargaFicheros	4	2	Pequeño
UtilesFichero	0	5	Pequeño
XMLParser	3	2	Pequeño

Tabla 21 Tamaño operacional de cada clase.

En las Figuras 9, 10 y 11 se muestran como quedan reflejados los atributos de calidad antes mencionados en las clases sometidas a la métrica de diseño tamaño operacional de clases. Consulte la Tabla 52 de los anexos para una información más detallada:

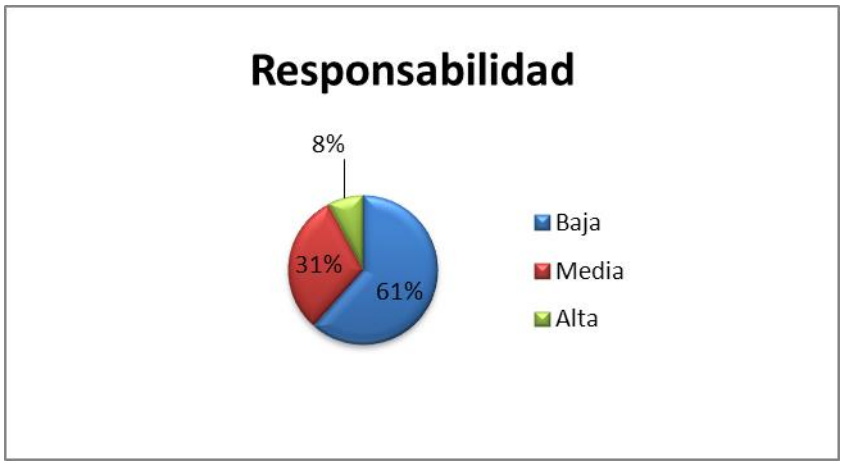

Figura 9 Atributo de calidad que refleja la responsabilidad.

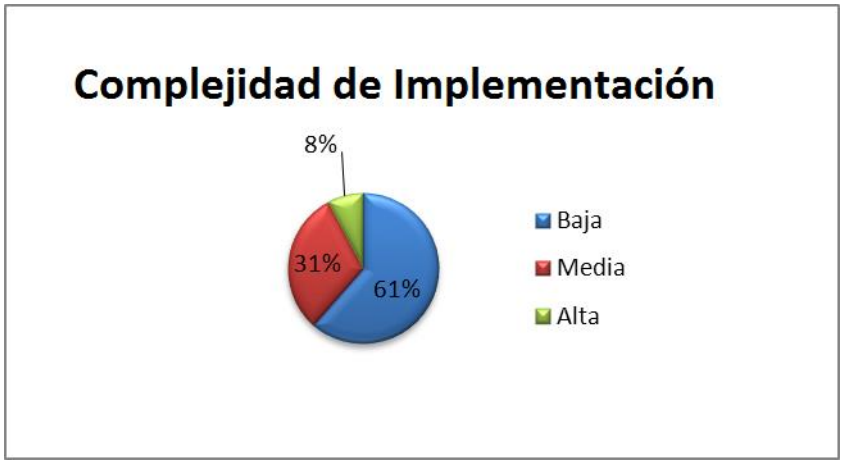

Figura 10 Atributo de calidad que refleja la complejidad.

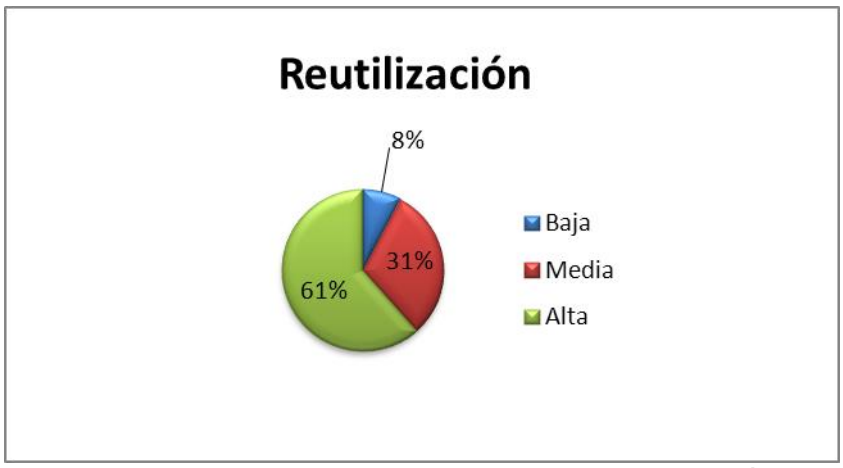

Figura 11 Atributo de calidad que refleja la reutilización.

3.3.1.2.1.1. Resultados obtenidos

Tomando como referencia la información referida por las Figuras 9, 10 y 11, donde se muestra con un 61% de umbral bajo como el porciento para los atributos de calidad responsabilidad y complejidad, y con un 61% de umbral alto como el porciento para el atributo de calidad reutilización, se concluye que los porcientos de responsabilidad y complejidad bajos obtenidos indican que más de la mitad de las clases, de manera general, no realizan un gran número de responsabilidades dentro del sistema y sus diseños son fáciles de implementar, solo un 31% de estas mostraron valores medios para dichos atributos, no siendo así con el porciento de reutilización obtenido, que muestra valores altos indicando que más de la mitad de las clases son altamente reutilizables dentro del diseño del sistema y que solo un 31% de estas mostraron valores medios de dicho atributo.

3.3.1.2.2. Relaciones entre clases (RC)

Está dada por el número de relaciones de uso de una clase con otra. Para medir el diseño según RC, se evalúan un conjunto de atributos de calidad entre los que se encuentra la Reutilización, definida en la métrica TOC, además los que se relacionan a continuación:

Acoplamiento: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.

Complejidad del mantenimiento: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.

Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado. (22)

Para determinar el grado de afectación de los atributos de calidad que mide la métrica RC es necesario determinar la cantidad de relaciones de uso (CRU) que posee cada una de las clases a medir. Una vez determinada la CRU, se procede a calcular el promedio de las mismas y teniendo ambos valores, según los criterios expuestos en las Tablas 22, 23, 24 y 25, se determina la incidencia de los atributos de calidad en cada una de las clases.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0

	Bajo	1
	Medio	2
	Alto	>2

Tabla 22 Rango de valores para medir la afectación del atributo acoplamiento

Atributo	**Categoría**	**Criterio**
Complejidad Mantenimiento	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Tabla 23 Rango de valores para medir la afectación del atributo mantenimiento

Atributo	**Categoría**	**Criterio**
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.

Tabla 24 Rango de valores para medir la afectación del atributo reutilización.

Atributo	**Categoría**	**Criterio**
Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Tabla 25 Rango de valores para medir la afectación del atributo pruebas.

En las Figuras 12, 13, 14 y 15 se muestran como quedan reflejados los atributos de calidad antes mencionados en las clases sometidas a la métrica de diseño relación entre clases. Consulte la Tabla 53 de los anexos para una información más detallada:

Figura 12 Atributo de calidad que refleja el acoplamiento.

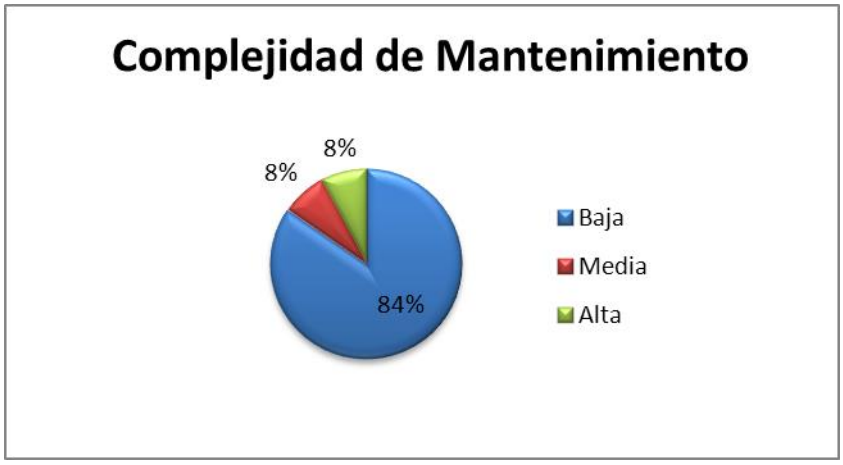

Figura 13 Atributo de calidad que refleja la complejidad de mantenimiento.

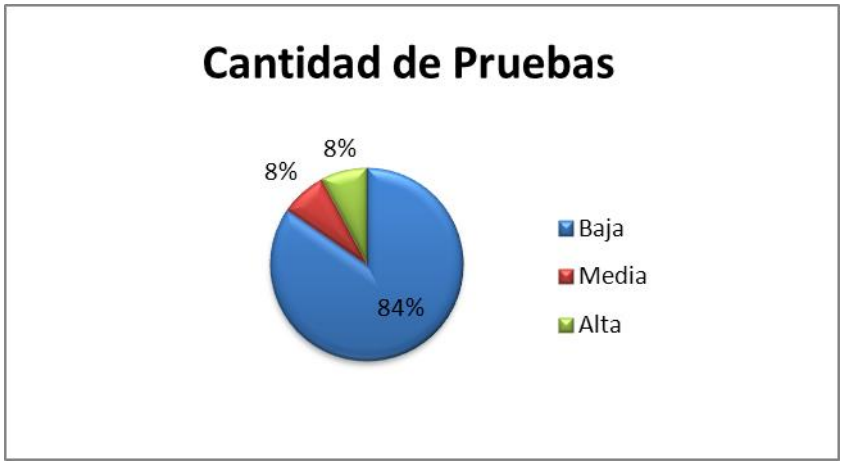

Figura 14 Atributo de calidad que refleja la cantidad de pruebas.

Figura 15 Atributo de calidad que refleja la reutilización.

3.3.1.2.2.1. Resultados obtenidos

Tomando como referencia la información referida por las Figuras 12, 13, 14 y 15, donde se muestra con un 46% de umbral nulo como el porciento para el atributo de calidad acoplamiento, con un 84% de umbral bajo como el porciento para los atributos de calidad complejidad de mantenimiento y cantidad de pruebas, y con un 84% de umbral alto como el porciento para el atributo de calidad reutilización, se concluye que el porcentaje nulo de acoplamiento obtenido indica que la mayor parte de las clases no dependen unas de otras, solo un 38% de estas muestran valores bajos de acoplamiento. Por otra parte, los porcientos bajos de complejidad de mantenimiento y cantidad de pruebas obtenidos, indican que no es necesario un alto grado de esfuerzo para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software así como tampoco es necesario realizar un gran esfuerzo a la hora de la realización de las pruebas de calidad al producto diseñado. Además se obtuvo un porciento alto de reutilización, lo que significa que más de la mitad de las clases son altamente reutilizables dentro del diseño del sistema.

3.3.2. Pruebas funcionales

A continuación se muestran los casos de prueba funcionales diseñados para cada una de las historias de usuario anteriormente mencionadas, debido a que las mismas incluyen un conjunto de funcionalidades críticas para el desarrollo exitoso del sistema informático propuesto en la investigación. Los restantes casos de pruebas están reflejados en los anexos.

Casos de prueba correspondientes a la funcionalidad: Crear el servicio de ejecución del componente de actualizaciones, encargada de ejecutar el componente de actualizaciones al

iniciar el sistema.

Caso de prueba de aceptación	
Código: HU1_CP1	**Historia de Usuario:** 1
Nombre: Crear el servicio de ejecución del componente de actualizaciones.	
Descripción: Prueba para la funcionalidad crear el servicio encargado de ejecutar el componente de actualizaciones con privilegios de administración.	
Condiciones de Ejecución: • El usuario debe ejecutar el instalador de la aplicación con privilegios de administración.	
Resultados esperados: En dependencia del sistema operativo el instalador de la aplicación debe crear el servicio o demonio encargado de ejecutar el componente de actualizaciones al iniciar el sistema.	
Evaluación de la prueba:	

Tabla 26 Caso de prueba: Crear el servicio de ejecución del componente de actualizaciones.

Caso de prueba de aceptación	
Código: HU1_CP2	**Historia de Usuario:** 1
Nombre: Crear el servicio de ejecución del componente de actualizaciones.	
Descripción: Prueba para la funcionalidad crear el servicio encargado de ejecutar el componente de actualizaciones sin privilegios de administración.	
Condiciones de Ejecución: • El usuario debe ejecutar el instalador de la aplicación sin privilegios de administración.	
Resultados esperados: Se muestra un mensaje indicando que no pudo ser creado el servicio de actualizaciones.	
Evaluación de la prueba:	

Tabla 27 Caso de prueba: Crear servicio de ejecución del componente de actualizaciones. Flujo alterno.

3.3.2.1. Análisis de los resultados

Para validar que la salida emitida por el sistema informático concordara con el resultado esperado por el cliente se diseñaron 13 pruebas funcionales en conjunto cliente-desarrolladores, de las cuales, en una primera iteración, de detectaron un total de 3 no conformidades. En una segunda iteración de las pruebas, fueron parcialmente resueltas las no conformidades identificadas durante la primera iteración quedando solo una sin resolver, por lo que fue necesario realizar una tercera iteración, obteniendo en esta un resultado satisfactorio para todas las pruebas realizadas.

A continuación se muestra como quedan reflejados los resultados por cada una de las iteraciones de pruebas funcionales realizadas el sistema:

Iteración	No conformidades detectadas	Total de pruebas realizadas
1	3	5
2	1	5
3	0	5

Tabla 28 Resultados para cada una de las iteraciones de las pruebas funcionales.

3.3.3. Pruebas unitarias

A continuación se muestran las pruebas unitarias aplicadas al sistema informático propuesto.

3.3.3.1. Prueba del camino básico

La prueba del camino básico es una técnica de prueba unitaria que permite al equipo de desarrollo obtener una medida de la complejidad lógica del código implementado como resultado del sistema propuesto en la investigación; usando dicha medida como guía para la definición de un conjunto de caminos independientes[3] de ejecución, lo que garantiza que durante la prueba se ejecuten por lo menos una vez cada sentencia del programa. (22)

3.3.3.1.1. Complejidad ciclomática

La complejidad ciclomática está basada en la teoría de grafos y da una métrica del software extremadamente útil, calculándose de tres formas:

1. El número de regiones del grafo de flujo debe coincidir con la complejidad ciclomática.

2. La complejidad ciclomática, V (G), de un grafo de flujo G se define como V (G)=A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

3. La complejidad ciclomática, V (G), de un grafo de flujo G también se define como V (G) P+ 1, donde P es el número de nodos predicado contenidos en el grafo de flujo G.

En la Figura 4 se puede apreciar el código correspondiente al método actualizar perteneciente a la clase ProcesoActualizacion que debido a la relevancia que posee en el sistema informático propuesto se le ha aplicado la métrica de complejidad ciclomática, donde se procede a:

[3] Camino independiente se entiende aquel que introduce un nuevo conjunto de sentencias o una nueva condición. En términos del grafo, por una arista que no haya sido recorrida antes.

- Confeccionar el grafo de flujo correspondiente.
- Calcular la complejidad ciclomática asociada.
- Extracción de los caminos independientes según el valor de la complejidad ciclomática.
- Realizar los casos de pruebas.
- Analizar los resultados obtenidos en las pruebas.

```
public void actualizar() throws IOException, JDOMException, InterruptedException {
    recuperacion();
    iconoSystemTray.displayMessage("Información", "Buscando actualizaciones disponibles de la aplicación.", TrayIcon.MessageType.INFO);
    con.descargarFichero("update.xml", "update.xml");
    actualizacion = xml.getDatosActualizacion("update.xml");
    if (!compararVersion(conf.getVersion(), actualizacion.getVersion())) {
        File f = new File("update.xml");
        f.delete();
        System.exit(0);
    }
    HiloDescargaFicheros thread = new HiloDescargaFicheros("descarga", con, actualizacion);
    thread.start();
    this.respaldo(actualizacion.getArchivos(), 0);
    thread.join();
    if (thread.Errores()) {
        JOptionPane.showMessageDialog(null, "Ocurrió un error mientras se descargaba la actualización del servidor.",
                "Actualizador SIRECC", JOptionPane.ERROR_MESSAGE);
        finalizarActualizacion();
    }
    crearDirectorios();
    this.actualizarFicheros(actualizacion.getArchivos(), 0);
    xml.setVersionCliente(actualizacion.getVersion());
    this.finalizarActualizacion();
}
```

Figura 16 Código del método actualizar

3.3.3.1.2. Grafo de flujo

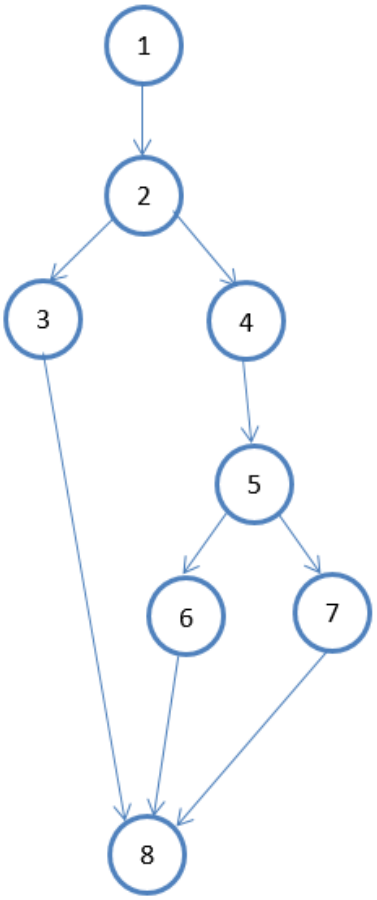

Figura 17 Grafo de flujo

3.3.3.1.3. Cálculo de la complejidad ciclomática:

1. El número de regiones del grafo de flujo debe coincidir con la complejidad ciclomática.

   V (G) = número de regiones
   V (G) = 3

2. La complejidad ciclomática, V (G), de un grafo de flujo G se define como V (G)=A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

   V (G) = A – N + 2
   V (G) = 9 – 8 + 2
   V (G) = 1 + 2
   V (G) = 3

3. La complejidad ciclomática, V (G), de un grafo de flujo G también se define como V (G) P+ 1, donde P es el número de nodos predicado contenidos en el grafo de flujo G.

   V (G) = P + 1
   V (G) = 2 + 1
   V (G) = 3

3.3.3.1.4. Caminos independientes establecidos.

Camino 1: 1-2-3-8
Camino 2: 1-2-4-5-6-8
Camino 3: 1-2-4-5-7-8

3.3.3.1.5. Casos de prueba.

Dando continuidad al método y contando con el grafo de flujo y los caminos independientes identificados, se procede a la creación de los casos de pruebas para cada uno de estos caminos. La tabla muestra el caso de prueba para el camino 1-3-5-6-19*,* los restantes casos de pruebas se encuentran reflejados en los anexos.

Casos de pruebas para el camino básico No. 1	
Camino	1-2-3-8
Descripción	Se llama al método recuperar actualización que chequea si el proceso anterior terminó correctamente comprobando la existencia del fichero de control de la actualización. Se descarga desde el servidor de actualizaciones el fichero update.xml con la información de la actualización existente en el servidor. Una vez descargado, se lee el fichero y se obtiene la versión de la actualización, los directorios nuevos a ser creados y los ficheros que deberán ser actualizados. Se compara la versión de la actualización con la de la aplicación instalada en el cliente y si no es necesario actualizar, se elimina el fichero descargado y se termina la ejecución del componente.
Entrada	Actualiza la versión de la aplicación instalada

	en el cliente en caso de ser menor que la de la actualización.
Resultado esperado	No actualizar la aplicación.

Tabla 29 Caso de prueba para el camino #1

3.3.3.1.6. Resultados de las pruebas.

Luego de terminada la fase de pruebas se identificaron un conjunto de no conformidades en la aplicación. Por lo que fue necesario realizarle una segunda iteración de pruebas al sistema, donde se obtuvieron todos los resultados satisfactorios. Finalmente se pudo comprobar que la aplicación cumple con los requisitos planteados al inicio del desarrollo.

3.3.4. Validación de la solución

Partiendo de que la solución informática, para lo cual fue creado el actualizador automático se encuentra en fases de desarrollo es imposible para los autores demostrar en un ambiente real el cumplimiento del objetivo planteado. En su lugar se montó un entorno de prueba compuesto por 4 PC con las siguientes características.

PC	Microprocesador	Memoria RAM	Tarjeta de red
1	Intel Core2Duo 2.2GHz	1Gb	Fast Ethernet 100/10 Mbps
2	Intel Core2Duo 2.2GHz	2Gb	Fast Ethernet 100/10 Mbps
3	Intel Pentium IV 3.0 GHz	1Gb	Fast Ethernet 100/10 Mbps
4	Intel Core2Duo 1.7GHz	3Gb	Gigabit Ethernet 1000/100/10 Mbps

Tabla 30 Características de cada PC

Las pruebas realizadas se centraron en la obtención del tiempo consumido en la actualización de una versión de SIRECC por cada PC de manera individual, para analizar el tiempo de actualización entre del método tradicional y la solución propuesta. Igualmente se realizaron las mismas pruebas pero para el conjunto con el objetivo de analizar los tiempos entre el proceso tradicional en su conjunto y la solución propuesta en el entorno concurrente propuesto, considerando que todas las estaciones se encienden al mismo tiempo y con un solo especialista.

Los resultados obtenidos en la medición de los tiempos de actualización para el proceso realizado de forma manual están sobre los 2 minutos en la mayoría de los casos, solo una PC

estuvo por debajo de los 2 minutos, solamente por 4 segundos. Por su parte en la medición de los tiempos de la actualización a través de la solución propuesta el tiempo promedio estuvo entre 1 minuto y 1:30 minutos, con la excepción de una de las PC que estuvo próximo a los 2:49 minutos. Es importante destacar que los mayores tiempos se obtuvieron de la PC con las menores prestaciones dentro de las pruebas. La siguiente tabla muestra los tiempos exactos obtenidos en las pruebas anteriores.

	Distribución de los tiempos de actualización de forma: (minutos : segundos : milisegundos)	
PC	**Manual**	**Automática**
1	2: 07: 89	1: 00: 60
2	2: 00: 22	1: 33: 72
3	2: 53: 57	2: 49: 57
4	1: 56: 76	0: 57: 31

Tabla 31 Tiempos obtenidos en las pruebas, de manera individual para cada PC.

A través del procedimiento manual a las 4 PC utilizadas para las pruebas se pudo obtener que el mismo tardó 8 minutos, 58 segundos y 44 milésimas de segundo en actualizar todas las estaciones de trabajo, eso sin tener en cuenta el tiempo necesario para trasladarse de una a otra. Para las mismas pruebas a través de la solución propuesta se completó la actualización en solo 2 minutos, 26 segundos y 63 milésimas de segundo.

Los resultados obtenidos para las condiciones antes descritas permiten arrojar que de manera independiente en las 4 PC se disminuyó el tiempo en al menos 4 segundos. De manera similar sucede para las pruebas realizadas en todas las PC tal cual sería el proceso permitiendo obtener, a través de la solución propuesta, un tiempo casi igual a la cuarta parte del tiempo obtenido a través del proceso manual. Una vez realizado el análisis de los tiempos obtenidos en las pruebas se puede asegurar que, al menos para el entorno similar al de prueba, la solución contribuye a la disminución del tiempo en las actualizaciones que se le realicen a SIRECC.

Teniendo en cuenta los resultados obtenidos para el entorno de pruebas antes descrito, y la distribución física de las estaciones de trabajo en la Cámara de Comercio, promediando el tiempo de ejecución manual que demora un técnico en realizar las actualizaciones ordenador por ordenador se obtiene un valor medio de tiempo de 2 minutos con 13 segundos,

multiplicando ese resultado por 25 que representa la cantidad de ordenadores que posee la Cámara de Comercio se obtiene un valor de tiempo de 55 minutos con 49 segundos, a este resultado se le suma el tiempo que demora el técnico en trasladarse por los 3 pisos que tiene la Cámara de Comercio que es de aproximadamente 3 minutos mas el tiempo que demora en ir de un local a otro que es de aproximadamente de 3 minutos, se obtiene como resultado final una demora total del proceso manual de 1 hora con 4 minutos y 49 segundos. Comparando este resultado final obtenido con el tiempo que tarda el componente desarrollado en realizar dicho proceso de forma automática que es de solo 2 minutos con 26 segundos, se observa una reducción considerable de dicho proceso en 1 hora y 2 minutos aproximadamente.

3.4. Conclusiones parciales del capitulo

En este capítulo se determinó, de acuerdo a las potencialidades y características de cada uno de los elementos correspondientes a las fases de implementación y pruebas antes descritos, la utilización de las métricas Estabilidad, Especificidad y Grado de validez, para validar los requisitos identificados, así como el uso de las métricas Tamaño Operacional de las Clases y Relaciones entre Clases para validar el diseño seleccionado. Además se diseñaron y aplicaron los casos de pruebas a cada una de las funcionalidades del sistema, y se montó un entorno de pruebas para comprobar la puesta en funcionamiento del componente arrojando como resultado una disminución considerable en el tiempo de actualización de las aplicaciones SIRECC y que todas las estaciones clientes se ejecutaran bajo la misma versión una vez concluido el proceso de actualización.

Conclusiones Generales

Con la culminación de este trabajo se obtuvieron resultados favorables que pueden concluirse de la siguiente forma:

- El estudio realizado de otras aplicaciones para resolver problemas similares planteó la necesidad de construir un sistema propio que cumpla con las necesidades planteadas por el cliente para realizar el proceso de actualizaciones en la Cámara de Comercio de la República de Cuba, de manera que fue necesaria la selección de herramientas, tecnologías y metodología adecuadas para conducir el desarrollo del sistema informático.
- Se implementó el sistema propuesto sobre la base de las funcionalidades identificadas por el usuario en conjunto con el equipo de desarrollo obteniéndose una versión funcional del sistema.
- Se realizaron las diferentes pruebas para la validación de la solución informática a través de las cuales se pudieron detectar y solucionar los errores en cada una de las iteraciones previstas por el equipo de desarrollo adecuándose el resultado obtenido a un alto grado de calidad.

Como resultado del proceso de desarrollo se obtuvo un producto para llevar a cabo las actualizaciones automáticas en la Cámara de Comercio de la República de Cuba que disminuye el tiempo de ejecución de esta tarea y la diferencia de versiones de la aplicación SIRECC. El sistema obtenido además cumple con los requisitos establecidos por el cliente y los estándares de calidad.

Recomendaciones

- Se recomienda realizar las pruebas funcionales del componente en un entorno con ordenadores que tengan el sistema operativo Linux instalado en cualquiera de sus distribuciones.
- Realizar las pruebas de funcionamiento del componente en el entorno real de la Cámara de Comercio, donde será puesto en práctica posteriormente.

Referencias Bibliográficas

1. Cámara de Comercio de la República de Cuba. [En línea] [Citado el: 18 de enero de 2013.] http://www.camaracuba.cu.

2. Code Project. [En línea] [Citado el: 24 de enero de 2013.] http://www.codeproject.com/Articles/9566/Updater

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

4. Corporation, Symantec. Symantec. Symantec. Symantec Corporation, 1995 - 2011. [En línea] [Citado el: 17 de enero de 2013.] http://www.symantec.com/index.jsp

5. Creative Commons Attribution 3.0 United States License. 2010. Nabber.org. Nabber.org. [En línea] CC-GNU GPL, 18 de febrero de 2010. [Citado el: 10 de enero de 2010.] http://www.nabber.org/projects/appupdater/

6. Microsoft Developer Network. MSDN. [En línea] [Citado el: 15 de febrero de 2013.] http://msdn.microsoft.com/es-es/library/t71a733d%28v=vs.80%29.aspx.

7. GROUSSARD, Thierry. Visual Basic. NET (VB. NET)-Programe con Visual Studio 2008. Ediciones ENI, 2009.

8. SIGHO MANUAL TÉCNICO 1.0 ID: FMT-A5-007 Instalador Inicial Guía de Instalación Versión del Módulo 3.0.0 Introducción al SIGHO Sesión II. Lic. Aarón García López

9. Martínez Cabrera, Jorge Ernesto y Milián Rodriguez, Rasiel. Mecanismo de actualización para el Sistema de Gestión Penitenciaria Venezolano. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. Habana, Cuba: UCI, Julio de 2007.

10. Ben Collins-Sussman, Brian W. Fitzpatrick, y C. Michael Pilato. Control de versiones con Subversion. 2002-2004.

11. Universidad Nacional Mayor de San Marcos. [En línea] [Citado el: 28 de enero de 2013.] http://www.unmsm.edu.pe/ogp/ARCHIVOS/Glosario/inds.htm.

12. Lapuente, María Jesús Lamarca. Tesis Doctoral Hipertexto: El nuevo concepto de documento en la cultura de la imagen. [En línea] [Citado el: 25 de enero de 2013.] http://www.hipertexto.info/

13. Universidad de Jaén. Descripción del Servicio de Transferencia de ficheros. Conceptos Básicos. [En línea] [Citado el: 25 de enero de 2013.] http://www.ujaen.es/sci/redes/ftp/concepto.html

14. Real Academia Española. Diccionario de la Lengua Española - Vigésima segunda edición. [En línea] [Citado el: 25 de enero de 2013.] http://lema.rae.es/drae/?val=lenguaje.

15. Java. [En línea] [Citado el: 25 de enero de 2013.] http://www.java.com/es/download/faq/whatis_java.xml.

16. Instituto Tecnológico de Veracruz. [En línea] [Citado el: 26 de enero de 2013.]

http://www.prograweb.com.mx/pweb/0102http.html.

17. Jacobson, Ivar y Booch, Grady. El Proceso Unificado de Desarrollo de Software. Pearson Addisson-Wesley. 2000.

18. Martinez, Alejandro. Guia a rational Unified Process. Castilla de la Mancha : s.n., 1997.

19. Ciencia y Técnica Administrativa (C&TA). [En línea] [Citado el: 29 de enero de 2013.] http://www.cyta.com.ar/ta0502/v5n2a1.htm.

20. Programación Extrema. [En línea] 2002. [Citado el: 1 de febrero de 2013.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf.

21. La nueva Metodología. [En línea] marzo de 2003. [Citado el: 1 de febrero de 2013.] http://www.programacionextrema.org/articulos/newMethodology.es.html.

22. S. Pressman, Roge. Ingeniería del Software. Un enfoque práctico. 2002.

23. NetBeans. [En línea] [Citado el: febrero 1, 2013.] http://netbeans.org/features/index.html.

24. Böck, Heiko. The Definitive Guide to NetBeans Platform. 2009.

25. Noble, Angela Martin y de Robert Biddle, James. The XP Customer Role in Practice: Three Studies Agile. 2004.

26. Bikha, Avinash . Requirements Management – Defining the Project Scope and Developing Use Cases. 2008.

27. ALEGSA. [En línea] [Citado el: 27 de enero de 2013.] http://www.alegsa.com.ar/.

28. Clements, Paul. 1996. A Survey of Architecture Description Languages. 1996.

29. Universidad de las Américas Puebla. UDLAP. [En línea] [Citado el: 20 de Febrero de 2013.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf.

30. Babylon. [En línea] [Citado el: 11 de marzo de 2013.] http://diccionario.babylon.com/bytecode/.

31. Oracle. [En línea] [Citado el: 11 de marzo de 2013.] http://www.oracle.com/technetwork/java/jms/index.html.

32. Larman, Craig. 2003. UML y Patrones. 2. 2003.

33. CVS. Concurrent Version System. [En línea] [Citado el: 16 de febrero de 2013.] http://www.cvshome.org/.

34. Commons Net. [En línea] [Citado el: 3 de 4 de 2013.] http://commons.apache.org/proper/commons-net/.

35. Real Academia Española. 2010. Vigésima segunda edición diccionario de la lengua española. Diccionario de la Lengua Española. [En línea] 2010. [Citado el: 12 de Enero de 2013.] http://lema.rae.es/drae/?val=actualizador.

36. Microsoft TechNet. Windows Server. [En línea] [Citado el: 24 de enero de 2013.] http://technet.microsoft.com/es-es/library/cc731604.aspx.