UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Herramienta de medición integral de parámetros de calidad de servicio mediante el protocolo SNMP en una red de área local.

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

Autores: José Ricardo Arias Pérez
Tutores: Dra. C. Marely del Rosario Cruz Felipe
Lic. Jenisley Verde Acosta

La Habana, junio 2014

Año 56 de la Revolución.

AGRADECIMIENTOS

A mis padres y abuelos por ser el apoyo fundamental en mi vida y mi educación. Por enseñarme que el esfuerzo y el sacrificio siempre trae las mejores recompensas.

A mis tíos por sus consejos siempre oportunos y por apoyarme a mí y a mi familia en estos tiempos difíciles.

A mis tutores Marely y Jeni por sus consejos y exigencias durante todo el proceso de elaboración de la tesis.

A mis amigos y hermanos, Adrián, Carlos, Etián, Karel, Carlos Rafael, Emmanuel, Richard y demás, los cuales hemos compartido cinco años de nuestras vidas.

Al Movimiento de Programación Competitiva – Tomás López Jiménez (MPC-TLJ) en el cual aprendí a pensar diferente y a estudiar mejor.

DEDICATORIA

A mis padres, José Ricardo y Belkis.

A mi familia y amigos.

DECLARACIÓN JURADA DE AUTORÍA

Declaro ser autor 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 _____.

_____

Firma de Autor

_____ _____

Firma de Tutor
Lic. Jenisley Verde Acosta

Firma de Tutor
Dra. C. Marely del Rosario Cruz Felipe

"Por lo general, aquello que más deseas no suele caerte encima; cae en algún sitio a tu alrededor, y tú tienes que darte cuenta de que está ahí y tienes que levantarte y que invertir el tiempo y esfuerzo necesarios para conseguirlo."

Neil Strauss Darrow

DATOS DE CONTACTO

Tutor: Marely del Rosario Cruz Felipe

Ciudadanía: cubana

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Doctora en Ciencias Técnicas, Máster en Telemática

Categoría Docente: Profesora Titular

E-mail: marely@uci.cu

Tutor: Jenisley Verde Acosta

Ciudadanía: cubana

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Licenciada en Educación Especialidad Informática

Categoría Docente: Profesora Asistente

E-mail: jeni@uci.cu

RESUMEN

Las tecnologías de la información constituyen actualmente el principal medio de comunicación existente en el mundo. Las redes se han convertido en un factor clave en los negocios, la ciencia y la sociedad, por lo que la disponibilidad de los servicios en la misma es crucial. La principal vía de optimizar el funcionamiento de las redes es a través del constante monitoreo y control de variables e indicadores, los cuales ayudan a determinar la calidad de los servicios que brindan las mismas. Muchas soluciones existentes actualmente permiten medir estos parámetros de manera eficiente y precisa, sin embargo, no abarcan todos los parámetros requeridos en esta investigación.

Debido a las deficiencias detectadas en las herramientas analizadas durante la investigación realizada, el presente trabajo brinda un ambiente sencillo e intuitivo el cual reúna todos los parámetros de calidad de servicio que se desean medir, usando el protocolo SNMP. La herramienta desarrollada permite reducir el esfuerzo necesario para medir los parámetros a los usuarios finales. El desarrollo de la herramienta está sustentado por una metodología ágil, XP, utilizando las técnicas de modelación establecidas por el Lenguaje Unificado de Modelado (UML), y apoyándose en el marco de trabajo Qt, finalizando con una serie de pruebas que validaron su calidad.

Palabras claves: calidad de servicio, redes LAN, SNMP.

ABSTRACT

The information technologies are currently the primary means of communication existing in the world. Networks have become a key factor in business, science and society, so that the availability of services in itself is crucial. The main way to optimize the operation of networks is through constant monitoring and control of variables and indicators, which help to determine the quality of services provided by them. Many existing solutions currently allow to measure these parameters accurately and efficiently, however, do not cover all the required parameters in this investigation.

Due to the deficiencies in the tools discussed during the investigation, this paper provides a simple and intuitive environment that covers all QoS parameters of a LAN, using the SNMP protocol. The developed tool reduces the effort needed to measure the parameters to end users. The development of the tool is supported by an agile methodology, XP, using the techniques of modeling defined in the Unified Modeling Language (UML), and based on the Qt framework work, ending with a series of tests that validated its quality.

Keywords: LAN, quality of service, SNMP.

ÍNDICE

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) han jugado un papel primordial en la evolución de la distribución de la información a lo largo y ancho del mundo, pues han revolucionado desde las formas más primitivas de comunicación hasta alcanzar el desarrollo informático actual.

Han facilitado el intercambio de información, en el contexto de las redes, brindando múltiples servicios con notables facilidades y beneficios para el desarrollo de las nuevas tecnologías. Entre estos servicios se destacan: correo electrónico, transmisión de archivos, consulta a páginas web, comercio electrónico, procesamiento de video y audio en tiempo real, entre otros.

En la actualidad, el desarrollo de la tecnología de redes ha experimentado un gran salto cualitativo, ocasionando que la cantidad de recursos destinados para su administración sea cada vez mayor y sean más complejos, lo cual conlleva a la evolución conjunta de acciones destinadas a optimizar dicha administración. Debido a esto el análisis y monitoreo de redes se ha convertido en una labor importante para determinar el desempeño de una red.

La Calidad de Servicio (QoS, Quality of Service) permite, a través de su implementación en una red de comunicaciones, ofrecer un mejor servicio en ciertos flujos de información, elevando o limitando su prioridad al viajar a través de la red. [Prado 2009]

En una Red de Área Local (LAN) los parámetros de QoS más significativos son **latencia**, **caudal**, **pérdida de paquetes** y **disponibilidad** [Cruz et al. 2013]. A pesar de que en Cuba, como en el mundo no se le presta la suficiente importancia a la realización de estos análisis, la medición de estos parámetros de QoS es importante ya que le brinda información a los especialistas para la toma de decisiones pertinentes.

A nivel mundial la forma más utilizada para obtener información de la QoS con que se administra una red es a través del empleo de herramientas de medición.

Actualmente en el mercado existe una gran variedad de herramientas para la medición de parámetros QoS, algunas privativas y otras de código abierto; cada entidad tiene su red adaptada a sus necesidades y a pesar de que las herramientas existentes pueden constituir una solución a las necesidades de cada una, muchas de estas herramientas no miden todos los parámetros requeridos o son de licencia privativa; todo esto trae como consecuencia que:

- Haya que tener instaladas varias aplicaciones a la vez para utilizar de cada una de ellas una mínima porción de sus funcionalidades, y así obtener la información del estado de la red que se necesita medir, ya que todas no miden estos indicadores de QoS,

ocasionando pérdida de tiempo, además del trabajo engorroso que ocasiona el empleo de varias herramientas a la misma vez.

- La mayoría de las herramientas que miden muchos de estos parámetros son propietarias y su coste es elevado.

Partiendo de esta situación analizada se determina como **problema científico** de la investigación: ¿Cómo medir parámetros de QoS en una LAN de forma integral[1]?

Siendo el **objeto de estudio** de este trabajo: Medición de parámetros de red mediante el Protocolo Simple de Gestión de Red (SNMP, Simple Network Management Protocol).

Para dar respuesta al problema planteado, se ha propuesto como **objetivo general**: Desarrollar una herramienta de medición integral de parámetros de QoS en una LAN, a través del empleo del protocolo SNMP.

Cuyo **campo de acción** está enmarcado en: Herramienta de medición integral de parámetros de QoS en una LAN con el empleo del protocolo SNMP.

Como **idea a defender** se tiene: Haciendo uso de las variables del protocolo SNMP y relacionando estas con las expresiones de los parámetros de QoS en una LAN es posible obtener una herramienta para la medición integral de dichos parámetros.

Objetivos Específicos:

- Realizar el estudio del estado del arte de las diferentes herramientas de monitoreo de red que existen para determinar los parámetros de QoS que miden.
- Diseñar e implementar la herramienta de medición de parámetros de QoS en una LAN.
- Validar el correcto funcionamiento de la herramienta de medición de parámetros de QoS mediante las pruebas unitarias y pruebas de aceptación.
- Validar que los resultados que entrega la herramienta de medición son correctos, a partir de la comparación con otras herramientas.

Para el desarrollo de la investigación se utilizan los siguientes **métodos científicos**:

[1] *El término integral hace referencia a que la herramienta agrupará todos los parámetros de QoS de una LAN (latencia, disponibilidad, caudal y pérdida de paquetes) y realizará mediciones de los mismos usando las variables que brinda SNMP.*

Métodos Teóricos:

- ***Histórico – Lógico:*** Se utiliza en el estudio de las diferentes herramientas de monitoreo que existen en la actualidad, para determinar el uso, las ventajas y desventajas de las mismas y para determinar si el empleo de éstas es de utilidad para la medición de los parámetros de QoS que se requieren en la investigación.
- ***Analítico – Sintético:*** Para analizar y extraer de la documentación especializada los aspectos relacionados con los parámetros de medición de QoS, el estudio del uso y características del protocolo SNMP, facilitando la búsqueda de conceptos fundamentales para el desarrollo de la investigación.

Métodos Empíricos:

- ***Observación:*** Se utilizará en la etapa de identificación y análisis de la información que fundamente su contribución en el estudio del campo de acción. Se analizarán las variables de SNMP que se relacionan con los parámetros QoS que se desean medir.
- ***Entrevistas:*** Se utiliza para la comunicación con los clientes e implicados en el proceso de recopilación de información, con vista a esclarecer la finalidad del trabajo.

Como resultados de la investigación, se obtiene una herramienta que permite medir de forma integral los parámetros de QoS de una LAN, siendo de utilidad para todas las ramas de la sociedad que dependan de un soporte informático y en especial para la Universidad de las Ciencias Informáticas.

El siguiente trabajo tiene como propósito llevar a cabo un ciclo completo de desarrollo del producto propuesto a construir. En el **Capítulo 1** se describe la fundamentación teórica de la investigación a realizar, se realiza un análisis del estado del arte que permite el estudio del conocimiento acumulado, así como la descripción de la metodología de desarrollo, tecnologías y herramientas a utilizar en el trabajo. En el **Capítulo 2** se describen los requisitos que debe cumplir la herramienta, así como los modelos y diagramas utilizados en el desarrollo de la misma. El **Capítulo 3** y final, abarca todo lo relacionado con las pruebas realizadas a la aplicación.

CAPÍTULO 1: ESTADO DEL ARTE

Introducción

En este capítulo se explican los conceptos fundamentales de QoS, así como los diferentes parámetros de QoS que se pretenden medir con la herramienta. Se definen los aspectos importantes del protocolo SNMP usado para la gestión de dispositivos y servicios en red, así como el protocolo ICMP usado para determinar el tiempo de respuesta de los dispositivos que se están encuestando. Se realiza un estudio y valoración de las principales herramientas que existen en la actualidad para el monitoreo y análisis de parámetros de red, con el fin de determinar las principales ventajas y desventajas que poseen y parámetros que miden.

1.1 Calidad de servicio (QoS)

El término QoS, se define según la Unión Internacional de Telecomunicaciones en ITU E.800: "Efecto global de las prestaciones de un servicio que determinan el grado de satisfacción de un usuario al utilizar dicho servicio". [ITU 2008]

La relación entre los términos calidad y servicio produce la siguiente definición: una medición del grado de confianza con que la red se desempeña y un intento de definir las características y propiedades de servicios específicos. [Ferguson and Huston 1998]

En otras palabras, calidad de servicio define cuan bien se encuentra una red para proporcionar determinados servicios y que estos sean percibidos en un margen de tiempo aceptable por los usuarios.

1.2 Parámetros de calidad de servicio

Una red debe garantizar un cierto nivel de calidad de servicios para un nivel de tráfico definido por la entidad en cuestión. Existen parámetros de QoS que se pueden medir y controlar para determinar la calidad en la comunicación; en el contexto de las LAN, los parámetros más significativos son: pérdida de paquetes, latencia, caudal y disponibilidad. [Cruz et al. 2013]

1.2.1 Caudal

El caudal o uso del ancho de banda disponible, se define como el tráfico total de datos que es recibido con éxito por el nodo destino en un tiempo determinado. [Y.1540 2005]

En conexiones a Internet, el ancho de banda es la capacidad de información de datos que se puede enviar a través de una conexión de red en un período de tiempo dado. El ancho de banda se indica generalmente en bits por segundo (bps), kilobits por segundo (kbps), o

megabits por segundo (Mbps). Por otro lado, también se define como: la relación existente entre el tamaño de la trama y el tiempo necesario para su transmisión que coincide con el ciclo de la comunicación. [Cruz et al. 2013]

Los cálculos de estos parámetros se realizan en intervalos de tiempo, lo cual permite establecer las variaciones en el tiempo del flujo de datos que circula por la red.

De esta manera, el caudal queda definido como: [Cisco Systems 2005]

$$C = B_x - B_{x-1}$$

Donde:

- **C** es el caudal calculado en bps.
- **Bx** es la cantidad de bits de información recibido por el nodo, en el tiempo **x.**

A partir de aquí se puede llegar a la siguiente fórmula, la cual brinda el porciento de utilización del ancho de banda del total disponible en el nodo.

$$C = \frac{(B_x - B_{x-1}) \times 8 \times 100}{(\Delta x) \times V}$$

Donde:

- **C** es el caudal calculado en porciento.
- **Bx** es la cantidad de bits de información recibido por el nodo, en el tiempo **x.**
- Δ**x** es el lapso de tiempo transcurrido entre una encuesta y otra.
- **V** es la velocidad de la interfaz por la que circula el flujo de datos.

1.2.2 Latencia

Se denomina a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación, transmisión y procesamiento de paquetes dentro de la red. [Andrade and Sebastiá 2005]

La latencia viene dada por la siguiente fórmula: [Mesa 2003]

$$L = TR - TE$$

Donde:

- **L** es la latencia del sistema en milisegundos.
- **TR** es el instante de tiempo en que se recibió el paquete ICMP.
- **TE** es el instante de tiempo en que se envió el paquete ICMP.

O sea, es el tiempo que tarda un dato en estar disponible desde que se realiza su petición; mientras menor sea el valor de la latencia, mejor desempeño tiene la red.

1.2.3 Pérdida de paquetes

Este parámetro indica el número de paquetes que se pierden durante la transmisión [Y.1540 2005]. Tiene una influencia significativa en el comportamiento de otros parámetros de QoS como la latencia, el caudal y la disponibilidad.

La pérdida de paquetes ocurre cuando uno o más paquetes de datos que viajan en una red IP fallan en alcanzar su destino. La causa de la pérdida de paquetes es debido a varios factores, entre los que se puede nombrar, degradación de la señal al viajar por el medio, interconexiones de la red sobresaturadas, paquetes con error rechazados en el tránsito, falla en el hardware de la red, o rutinas normales de enrutamiento. [Flickenger et al. 2006]

Se define la pérdida de paquetes como: [Mesa 2003]

$$PP = P_x - P_{x-1}$$

Donde:

- **PP** es la pérdida de paquetes total en paquetes por segundo.
- $\mathbf{P_x}$ es la cantidad de paquetes que ha sido descartado por el sistema.

- **Px** se define como **paquetes descartados + paquetes erróneos + paquetes de protocolo desconocido**.

1.2.4 Disponibilidad

La disponibilidad indica la utilización de los diferentes recursos y se especifica en porciento. [Y.1540 2005]

En la práctica es una medida de la garantía de ejecución de los servicios a lo largo del tiempo, y depende de factores tales como:

- Disponibilidad de los equipamientos utilizados en la red propietaria (red del cliente).
- Disponibilidad de la red, para cumplir con determinados requisitos de QoS.

Para el cálculo de la disponibilidad se empleará la expresión definida a continuación: [Mesa 2003]

$$D = (Dx / Dt) \times 100$$

Donde:

- **D** es la disponibilidad en porciento.
- **Dx** es la cantidad de encuestas en las que se encontraba disponible el equipo.
- **Dt** es la cantidad total de encuestas realizadas al equipo.

Una vez analizados los parámetros de Qos para una LAN, los cuales serán medidos a partir de los cálculos que realizará la herramienta, es necesario analizar otros factores que influyen en la elaboración de la misma, lo cual se muestra a partir del siguiente epígrafe.

1.3 Gestión de redes

Las herramientas de medición de parámetros de red se pueden clasificar en dos grandes grupos, uno donde se encuentran herramientas que se basan en modelos y simuladores de redes y otro para las herramientas de medición de parámetros reales; en este último se encuentra enmarcado este trabajo y lo constituyen las herramientas de gestión de redes.

La gestión de redes incluye el despliegue, integración y coordinación del hardware, software y los elementos humanos para monitorizar, probar, sondear, configurar, analizar, evaluar y controlar los recursos de la red para conseguir los requerimientos de tiempo real, desempeño operacional y calidad de servicio a un precio razonable. [Saydam and Magedanz 1996]

Los componentes de la gestión de redes son: [UIT-T 1992]

- ***Gestión de configuraciones/cambios:*** Mantener información relativa al diseño de la red y su configuración actual: gestión de inventario (Registro de la topología, base de datos de elementos de la red, historia de cambios y problemas), control operacional de la red (iniciar/detener componentes individuales, alterar la configuración de los dispositivos, cargar y configurar versiones de configuraciones, actualizaciones de hardware/software).

- ***Gestión del desempeño/contabilidad:*** Garantizar unos niveles consistentes de rendimiento (colección de datos, estadísticas de interfaces, tráfico, tasas de error, utilización, disponibilidad porcentual, análisis de datos para mediciones y pronósticos, establecimiento de niveles límites de rendimiento).

- ***Gestión de fallas:*** Identificación, aislamiento, reacción y resolución de la falla (mecanismos de reporte, instalación y control de procedimientos de alarmas, sistema de manejo de incidencias).

- ***Gestión de seguridad:*** Controlar el acceso a los recursos de la red de acuerdo a políticas bien definidas (uso periódico de herramientas para analizar y controlar el uso legítimo de la red, distribución de certificados, sondeo de vulnerabilidades, análisis de trazas, filtros de servicios, cifrado).

La herramienta a desarrollar se encuentra ubicada en el área de la gestión del desempeño, que es la que permite:

- Medir datos o variables, indicadores de rendimiento (caudal de la red, los tiempos de respuesta o latencia, etc.).
- Fijar los niveles mínimos de rendimiento que pueden ser tolerados.
- Monitorear todos los recursos para detectar cuellos de botella y traspaso de los umbrales.
- Realizar mediciones y análisis de tendencias para predecir las fallas que puedan ocurrir.
- Procesar los datos medidos para elaborar reportes y gráficas de desempeño.

Existen diversos protocolos para la administración de las redes informáticas, entre los que cabe destacar: **CMIP** (Protocolo de Información de Administración Común desarrollado por la ISO), **TMN** (protocolo para la Administración de Redes de Telecomunicaciones, definido por la ITU-T para la gestión de sistemas abiertos en una red de comunicaciones), y **SNMP** (Protocolo Simple de Administración de Redes, desarrollado para el intercambio de datagramas en redes). El más difundido y utilizado en el mundo para las tareas de administración de redes es SNMP [AGÉ et

al. 2011], utilizado para realizar labores de monitoreo de redes y es capaz de entregar información requerida por los indicadores de desempeño. Los agentes de los equipos brindan gran cantidad de datos sobre el estado de la red y la información se actualiza en los equipos en tiempo real. El protocolo SNMP será usado en el desarrollo de la herramienta.

1.4 Especificaciones sobre el protocolo SNMP

El protocolo SNMP fue diseñado como una solución a los problemas de comunicación entre diferentes tipos de redes. SNMP es utilizado para comunicar información de administración entre estaciones y agentes. Las metas de la arquitectura del protocolo son las siguientes:

- Minimizar la cantidad y la complejidad de las funciones de administración, llevadas a cabo por los agentes. Esta meta es muy importante debido a que:
    - Reduce los costos en rendimiento del software asociado.
    - Aumenta el nivel de apoyo en la administración remota, propiciando un uso más completo de las tareas de administración.
    - Se imponen menos restricciones a las herramientas administrativas.
    - La simplicidad en las funciones de administración hace más fácil el entendimiento y la implementación para los desarrolladores de herramientas de gestión de redes.
- Lograr un proceso de monitoreo y control que sea extensible e incrementable.
- Lograr una completa independencia de la arquitectura de un determinado host o un determinado *gateway*[2], es decir, lograr independencia del hardware. [Oviedo and Inatty 2008]

1.4.1 Elementos de la arquitectura SNMP

Nodos administrados: Ejecutan agentes SNMP[3] y consola de administración. Los recursos administrados pueden ser enrutadores, servidores, conmutadores, entre otros.

---

[2] *Gateway: Una **pasarela, puerta de enlace** o **gateway** es un dispositivo que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación. Su propósito es traducir la información del protocolo utilizado en una red al protocolo usado en la red de destino.*

[3] *Agentes SNMP: Programas contenidos dentro de los dispositivos SNMP con el objetivo de responder las peticiones realizadas a los dispositivos o enviar notificaciones del estado de determinadas variables. Se encargan de actualizar, en tiempo real, las variables de estado de los dispositivos.*

SNMP: Método de comunicación entre los dispositivos administrados y los servidores. Proporciona un mecanismo para acceder a los objetos de MIB de modo que puedan ser consultados y modificados.

Base de información de administración (MIB): Una base de datos relacional (organizada por objetos o variables y sus atributos o valores) que contiene información del estado y es actualizada por los agentes. La MIB será definida con más detalles más adelante.

Entidad administradora: Encargada de supervisar y controlar los recursos administrados mediante la solicitud de información a los agentes, de establecer valores a variables en el dispositivo mediante órdenes de escritura y es la receptora de los eventos notificados por los agentes. [CABLETRON SYSTEMS et al. 1999]

A continuación se muestra la figura 1, la cual contiene una representación de lo descrito.

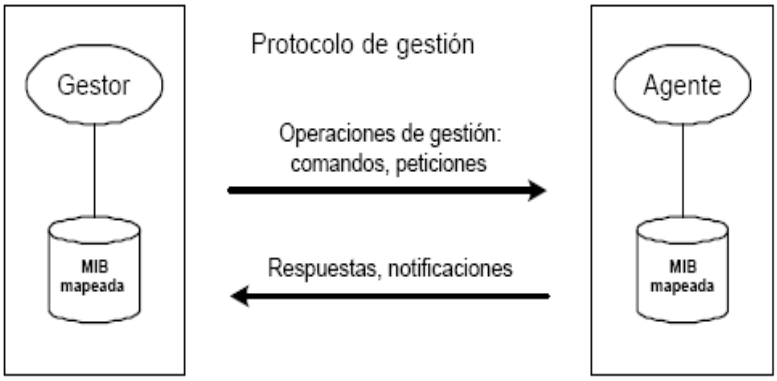

Figura 1: Arquitectura de un sistema de gestión SNMP.

En la figura se muestra una arquitectura SNMP en la cual están presentes el gestor y el agente; esta arquitectura se basa en el envío de peticiones desde el gestor hacia los agentes ubicados en los equipos, y estos se encargan de responder con los datos solicitados previamente, usando el protocolo SNMP como medio de comunicación.

1.4.2 Principales operaciones de SNMP

Las principales operaciones que se realizan con SNMP y que permiten la comunicación entre la MIB del agente y la estación administradora son:

- ***GET-REQUEST:*** Operación ejecutada por la estación administradora para obtener el valor de una o varias variables de la MIB de un dispositivo.

- ***GET-NEXT-REQUEST***: Operación ejecutada por la estación administradora para obtener el valor de una o varias variables de la MIB de un dispositivo. La diferencia con la operación anterior radica en que esta devuelve la próxima variable del árbol de la MIB.
- ***GET-BULK-REQUEST***: Operación ejecutada por la estación administradora para obtener el valor de una o varias variables de la MIB de un dispositivo de forma eficiente. Es muy usada para obtener los valores de tablas de gran tamaño.
- ***SET-REQUEST:*** Operación realizada por la estación administradora para establecer un valor sobre una o varias variables de la MIB en un dispositivo administrado.
- ***RESPONSE:*** Operación ejecutada por el agente SNMP en respuestas a las operaciones GET-REQUEST, GET-NEXT-REQUEST o SET-REQUEST para enviar los datos solicitados.

1.4.3 Base de información de administración (MIB)

La MIB es una base de datos, relacional y jerárquica, donde se almacenan los valores de diferentes variables importantes de los dispositivos. La misma está alojada en la memoria interna de los dispositivos, y es actualizada por los agentes en tiempo real.

La estructura de esta MIB es de árbol, donde cada nodo está compuesto por una etiqueta que lo nombra y un identificador numérico único continuo en el nivel de profundidad donde se encuentra; en caso de que el nodo sea hoja, entonces también estará el valor en el formato correspondiente al tipo de dato que represente. La MIB define un conjunto de variables estándares para todos los dispositivos administrados, la definición de dichas variables se encuentra registrada en el estándar *RFC1157*. Además, la MIB posibilita que las empresas que desarrollen dispositivos con características específicas enlacen sus variables de manera independiente. [Group 1990]

La MIB-2 es la rama estándar compatible con la mayoría de los dispositivos existentes actualmente y esta es la que será usada para el presente trabajo.

1.5 Protocolo ICMP

Para el cálculo de latencia en sistemas de monitoreo de redes se emplea el protocolo ICMP (**I**nternet **C**ontrol **M**essage **P**rotocol o Protocolo de Mensajes de Control de Internet) ya que es usado para la transmisión de mensajes de errores o mensajes del tipo *ICMP Query*; el mismo es muy empleado en conjunto con SNMP para el monitoreo de las redes.

Cada mensaje ICMP contiene tres campos que definen su propósito y proporcionan una suma de comprobación. Estos campos son tipo, código y suma de verificación. El campo **Tipo** identifica el mensaje ICMP, el campo de **Código** proporciona más información sobre el campo de tipo asociado y la **Suma de verificación** proporciona un método para determinar la

integridad del mensaje. Entre los tipos definidos se encuentran: *Echo Reply, Destination Unreachable, Echo Request, Time Exceeded.* El tipo *Echo Reply* es el más utilizado para probar la conectividad IP conocida comúnmente como **PING** de ICMP y es usado para medir los tiempos de respuesta y retardo en la conexión entre dos equipos. [Microsoft 2007]

A continuación se muestra la figura 2, la cual contiene el datagrama definido para el protocolo ICMP.

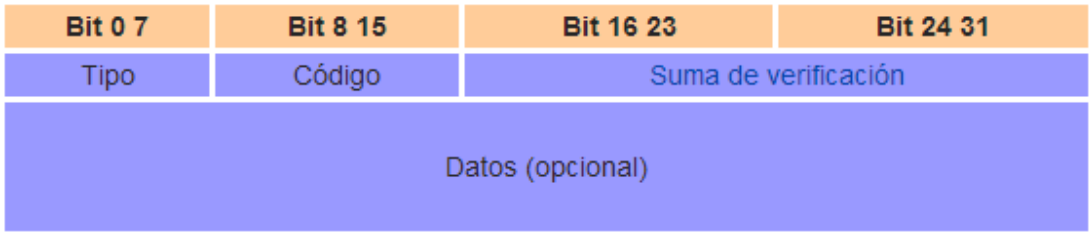

Figura 2: Datagrama del protocolo ICMP.

1.6 Herramientas de monitoreo de red

El mercado de la monitorización cuenta con múltiples plataformas que permiten un análisis de la infraestructura y la calidad de los servicios que la misma ofrece. A la hora de elegir un sistema de monitoreo de red se deben tener presente aspectos fundamentales como: precio, facilidad de instalación y de utilización, etc. Para el análisis de las herramientas existentes actualmente se van a tener en cuenta varios indicadores a evaluar los cuales se mencionan a continuación:

- Parámetros de QoS que miden.
- Posee licencia libre o es de código abierto.
- Uso del protocolo SNMP.
- Facilidad de manejo de la herramienta, y si posee interfaz gráfica.

1.6.1 Herramientas de monitoreo de red existentes

Actualmente, existe una gran variedad de herramientas de monitoreo de redes, las cuales brindan muchas funcionalidades y ventajas. Teniendo en cuenta los indicadores anteriores, se analizarán a continuación las más acordes a las necesidades planteadas para la investigación:

- **Multi Router Traffic Grapher (MRTG)**

Es una herramienta de código abierto, escrita en Perl y que utiliza SNMP para consultar a los dispositivos. Su instalación y configuración no es muy intuitiva, ya que está basado en scripts.

Es una herramienta para monitorizar la carga de tráfico sobre los nodos de una red, y almacena los resultados en una base de datos. Genera páginas HTML con gráficas sobre el uso de los dispositivos a través del tiempo.

MRTG fue concebido como un programa no diseñado para empresas en su momento. Su salida a Internet mostró la necesidad de este tipo de programas en el ámbito de las empresas. Es una solución ideal para ambientes que no requieran tanta complejidad y recursos. [Oetiker 1998]

Luego de la realización de pruebas y análisis a la herramienta, se determinó que la misma no brinda la medición de los parámetros latencia, pérdida de paquetes y disponibilidad directamente, sino que depende de scripts que hacen los administradores de la red u otros usuarios para hacer los cálculos de los mismos, usando las variables que brinda SNMP.

La herramienta no es de utilidad ya que no mide todos los parámetros de QoS que se requieren.

- **Cacti**

Cacti es una herramienta publicada bajo la licencia GPL[4] (GNU Public License). No dispone de una versión comercial ni de una licencia privativa para la distribución del producto, tampoco tiene soporte comercial o profesional. Pone a disposición un soporte gratuito mantenido por la comunidad mediante una lista de correos y un foro. La recolección de datos se realiza mediante el protocolo SNMP y se almacenan los resultados en una base de datos. Muestra gráficamente el estado de cada uno de los equipos, mediante el uso avanzado de gráficas. [Cruz 2005]

Al analizar esta herramienta, se pudo observar que la misma monitorea caudal en los dispositivos y pérdida de paquetes, mediante el uso del protocolo SNMP.
Aunque también realiza mediciones de disponibilidad, lo hace mediante el envío de paquetes ICMP y luego realiza cálculos para determinar la disponibilidad; mediante el uso de SNMP esta medición se hace más eficiente, ya que este maneja diferentes estados sobre las interfaces de red que permiten determinar la disponibilidad con más precisión.

Por otro lado, Cacti no funciona con agentes en los nodos, lo que provoca que cada uno tenga que ser configurado de manera distinta, en dependencia del sistema operativo. La configuración de los plugins no es una tarea trivial para usuarios sin experiencia, por lo que dificulta el uso de la aplicación.

Esta herramienta mide los parámetros de QoS requeridos, pero no lo hace de la manera ni con la precisión deseada, por lo que no cumple con los requisitos de la investigación.

- **Pandora FMS**

*[4] GPL: **G**eneral **P**ublic **L**icense o Licencia Pública General, fue creada por la Free Software Foundation en 1989, y está orientada principalmente a proteger la libre distribución, modificación y uso de software.*

Posee una versión *Open Source* [5] y otra privativa; esta última incluye además soporte profesional, actualizaciones y mantenimiento. Está orientado a la monitorización de hardware, software y aplicaciones. Puede supervisar gran cantidad de parámetros o servicios, mediante agentes específicos que recolectan información, incluso hasta sensores. Puede monitorizar mediante **SNMP**, **WMI**[6] y pruebas de red (**TCP/ICMP**) y comprobar cualquier sistema hardware con conectividad TCP/IP. Posee una buena escalabilidad y potencial para monitorizar una gran red de dispositivos. [Cruz 2005]

Como desventaja principal tiene que es una herramienta orientada a la monitorización del software y los servicios. No realiza mediciones del parámetro pérdida de paquetes, además que no es un sistema fácil de usar; es una herramienta versátil y potente que requiere cierta habilidad, conocimientos y experiencia previa en sistemas.

- **Whats Up**

WhatsUp ofrece monitoreo de aplicaciones y de redes, es fácil de utilizar y desplegar. Es un producto de la compañía ***Ip switch*** que permite monitorear la red, incluyendo servidores, servicios, dispositivos, etc., además que incluye soporte para monitoreo de redes en ambientes Windows.

WhatsUp contribuye a reducir caídas en la red y transforma datos aislados en información relevante para apoyar decisiones de negocios. Proporciona una implementación sencilla y se distribuye bajo licencia comercial.

WhatsUp es instalado con cinco monitores de rendimiento que monitorean tipos de datos específicos de los dispositivos de la red:

1. Utilización de CPU.
2. Utilización de Disco.
3. Utilización de Interfaz/Ancho de Banda.
4. Utilización de Memoria.
5. Latencia y Disponibilidad vía Ping.

---

[5] Open Source (Código Abierto) es la expresión con la que se conoce al software distribuido y desarrollado libremente. Destaca el acceso al código y la libre modificación del mismo por encima de las ganancias que genera el código privativo y las patentes de software. Se centra en que al compartir el código, el programa resultante tiende a ser de calidad superior al software propietario.

[6] WMI: Windows Management Instrumentation o Instrumental de Administración de Windows, es una iniciativa que pretende establecer normas estándar para tener acceso y compartir la información de administración a través de la red de una empresa.

La herramienta WhatsUp reúne muchos de los parámetros requeridos incluyendo la facilidad en su uso; a pesar de todo esto la misma posee licencia privativa por lo cual no cumple con los requerimientos establecidos.

- **OpenNMS**

OpenNMS es una herramienta de monitorización basada en software libre, está actualmente bajo la licencia GPL y es una de las herramientas más antiguas en el ámbito de la monitorización de redes, lo que le ha dado la posibilidad de nutrirse y perfeccionar a través de los años su funcionamiento. Ha alcanzado varios premios como *Inforworld Best of OpenSource Software* (BOSSIE), mejor software en la categoría de gestión de redes en los años 2009 y 2010 o *Best Systems Management Tool of LinuxWorld* en 2005. Actualmente está diseñada para soportar decenas de miles de nodos y utiliza SNMP para las notificaciones de eventos mediante alarmas. [Cruz 2005]

Entre las funcionalidades de OpenNMS se pueden destacar que es una herramienta capaz de auto-descubrir los servicios en la red en la cual está funcionando. El aprovisionamiento de los procesos es asíncrono para la escalabilidad, y existen redes de suministro con más de 50.000 dispositivos conectados.

También ofrece servicios para la gestión de los tiempos de respuesta, como pueden ser solicitudes ICMP (ping), visitas a un puerto determinado TCP para comprobar la instancia, aplicaciones web mediante el protocolo HTTP.

Como desventajas se tiene, que requiere un nivel muy elevado de conocimientos para su puesta en marcha y optimización del sistema, lo cual conlleva una serie de modificaciones y optimizaciones en la instalación, a nivel de configuración, hardware, base de datos, sistema operativo, etc.

Al igual que muchas herramientas, puede realizar mediciones de latencia usando el protocolo ICMP y enviando paquetes hacia la estación que se administra. También puede monitorear uso de ancho de banda configurando SNMP.

Según la investigación realizada no se encontró que realizara mediciones de pérdida de paquete ni disponibilidad hacia los dispositivos.

1.7 Herramientas y tecnologías

Para el desarrollo de la aplicación se utilizan diferentes herramientas y tecnologías, las cuales se describen a continuación:

- **Eclipse IDE:** Eclipse es una plataforma de desarrollo de código abierto hecha en Java, por lo que es multiplataforma; de por sí, es solo un marco de trabajo pero cuenta con diversos *plugins* que lo hacen muy extensible y adaptable a las necesidades de cada desarrollador. [Gallardo 2012]

  Se seleccionó esta herramienta ya que es integrable con aplicaciones de diseño, documentación y planificación, a la vez que brinda técnicas avanzadas de refactorización y análisis de código.

- **Qt Creator:** El *framework*[7] Qt fue publicado en el año 1995. Qt Creator es un IDE creado para el desarrollo de aplicaciones con las bibliotecas Qt, requiriendo su versión 4.x. Está disponible para los sistemas operativos Linux, Mac y Windows, permitiendo al desarrollador crear aplicaciones para múltiples sistemas o plataformas móviles. [Albán 2008]

  Qt Creator es una herramienta avanzada de desarrollo de aplicaciones. Brinda numerosas funcionalidades que facilitan el trabajo del programador:

  - Una interfaz amigable para el trabajo con el código: posee completamiento y refactorización de código, manejo de tabulaciones, resaltado de sintaxis, entre otros.

  - Posee una herramienta para desarrollar interfaces, llamada *Qt Designer*, con variedad de componentes que facilitan el trabajo del desarrollador.

  - Usa compilación multiplataforma, lo que permite la reutilización del código independientemente de la plataforma de trabajo.

Esta herramienta se seleccionó por su amplia gama de componentes visuales en su biblioteca gráfica, lo cual ayuda a la hora del desarrollo de las interfaces de la aplicación.

- **Visual Paradigm:** Visual Paradigm es un conjunto de herramientas CASE[8] para el modelado de diagramas UML[9], con soporte multiplataforma y licencia libre. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos,

---

[7] Framework: infraestructura digital, son diseñados con la meta de ayudar la tarea de los programadores, enfocándolos más al desarrollo del software que a los detalles de bajo nivel del funcionamiento de los programas. Es el esqueleto sobre el cual varios objetos son integrados para facilitar una solución dada.

*[8] CASE: **C**omputer **A**ided **S**oftware **E**ngineering o Ingeniería de Software Asistida por Computadora, son un conjunto de herramientas destinadas a reducir los costos de desarrollo de aplicaciones. Brindan ayuda al desarrollo del software en cualquiera de sus etapas, como diseño, planificación, cálculo de costos, etc.*

*[9] UML: **U**nified **M**odeling **L**anguage o Lenguaje Unificado de Modelado, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema; sirve para especificar y describir métodos o procesos.*

construcción, pruebas y despliegue. Permite diseñar muchos de los tipos de diagramas de clases, generar código desde diagramas y generar documentación. Esta herramienta se usa para hacer los diagramas necesarios en la etapa de diseño de la solución.

- **Biblioteca Net-SNMP:** Net-SNMP es una colección de herramientas relacionadas al protocolo SNMP incluyendo:

    - Un agente extensible.
    - Una biblioteca de desarrollo para aplicaciones.
    - Herramientas para encuestar información desde los agentes SNMP.
    - Herramientas para generar y manejar las alarmas SNMP.

  Este paquete está basado originalmente en la implementación de SNMP de la Universidad de Carnegie Mellon, pero ha sido desarrollado significativamente desde entonces. [Comunity 2011]

  Net-SNMP provee una implementación robusta y con una CLI (Interfaz de Línea de Comandos, por su acrónimo en inglés de **C**ommand **L**ine **I**nterface) sencilla de usar para el desarrollador. [Rockwood 2004]

  Esta biblioteca es un elemento clave para el desarrollo de esta aplicación, ya que es la encargada de establecer la comunicación con los dispositivos a nivel de la capa de transporte. La misma tiene gran prestigio en el mundo de SNMP y posee una interfaz de programación para C++, el cual es el lenguaje usado para el desarrollo.

- **Bibliotecas Boost:** El uso de bibliotecas de alta calidad como las de boost aceleran el desarrollo de aplicaciones y resulta en menor cantidad de errores y tiempo de mantenimiento del código. La biblioteca boost posee una tendencia hacia el estándar, por lo que muchos programadores están familiarizados con ella. Actualmente existen muchas aplicaciones que incluyen la boost, como es Adobe Acrobat Reader 7.0. [Dawes 2011]

  Esta biblioteca constituye otro elemento importante para el desarrollo de la aplicación, ya que se usa en conjunto con la net-snmp para asegurar la comunicación con los dispositivos. Además, se usa para el manejo y sincronización de los hilos de ejecución usados, así como el uso de peticiones asíncronas.

- **RRD Tools:** RRDtool (acrónimo de **R**ound **R**obin **D**atabase tool) es una herramienta con licencia GNU desarrollada por Tobias Oetiker, del Swiss Federal Institute of Technology.

A pesar de ser un tipo de base de datos, existen marcadas diferencias entre RRDtool y otros tipos: [Oetiker 2013]

- RRDtool almacena datos y al mismo tiempo brinda funcionalidades para la creación de gráficas. Otras bases de datos solo almacenan los datos.

- En las bases de datos comunes los nuevos datos son almacenados al final de la misma, por lo que se mantiene en crecimiento con el tiempo y a veces es necesario su mantenimiento. RRDtool se puede ver como una base de datos circular, en donde los datos antiguos son sobre escritos cuando se alcanza el límite de tiempo establecido, manteniéndose así con un tamaño constante. De aquí su nombre: base de datos "*Round Robin*".

- Las bases de datos comunes almacenan los valores como mismo son insertados, en cambio RRDtool puede calcular tasas de cambio, medias, máximos, y almacenar estos valores.

- Otras bases de datos son actualizadas cuando reciben los datos. RRDtool está estructurada de tal manera que necesita los datos en intervalos de tiempo predefinidos. Si no recibe los datos en ese tiempo, almacena un tipo especial llamado *UNKNOWN* para ese intervalo. Por este motivo es imperativo que la base de datos sea actualizada a cada intervalo.

Esta biblioteca brinda una base de datos para almacenar los valores históricos del proceso de monitoreo, así como la posibilidad de generar gráficas y reportes de valores.

1.8 Metodología a utilizar

La Programación Extrema (*XP*, siglas en inglés) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. [Sevilla and D.F 2010]

- Se utiliza en proyectos con corto tiempo de entrega, lo cual está presente en la herramienta.

- Es recomendable cuando existe un contacto directo y frecuente con el cliente, para satisfacer sus requisitos.

En la figura 3 se muestran las diferentes fases de desarrollo que posee XP: [Castillo and Sevilla 2010]

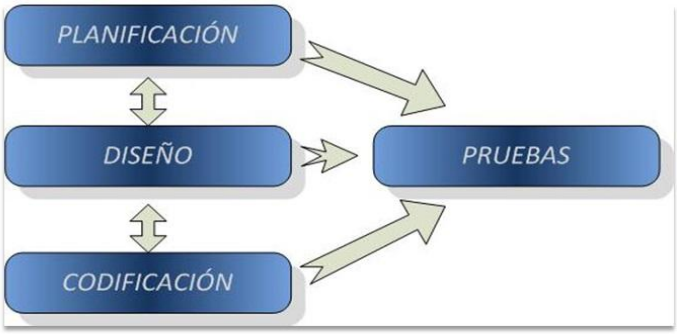

Figura 3: Fases de desarrollo según XP.

Esta metodología se adapta a todas las condiciones del sistema a desarrollar por lo expuesto anteriormente, por lo que se selecciona como la más idónea.

1.9 Conclusiones parciales

Las investigaciones realizadas dieron como resultado la necesidad de desarrollar la solución propuesta, debido a las deficiencias que poseen las herramientas existentes actualmente. Se observa la necesidad de que exista una herramienta para integrar todos estos parámetros de QoS.

Se evidencia la importancia del protocolo SNMP como principal vía para la medición de los parámetros de calidad y servicio requeridos, debido a su robustez y a la gran cantidad de información que almacena sobre los dispositivos. Además, se apoya en el protocolo ICMP para el cálculo de latencia hacia los dispositivos.

Luego del análisis de las herramientas y tecnologías, se determinó usar: una metodología ágil ***XP,*** como lenguaje de programación ***C++, Qt Creator*** y ***Eclipse*** como *IDEs* de desarrollo; se usará la biblioteca ***Net-SNMP*** como apoyo para la comunicación con los dispositivos, y ***RRDtool*** para almacenar las trazas y la generación de reportes y gráficas.

CAPÍTULO 2: PROPUESTA DE LA SOLUCIÓN

Introducción

En el presente capítulo se especifican las características de la propuesta de solución dada. Se propone una estructura de las funcionalidades del sistema a través de Historias de Usuarios (HU), que es el primer artefacto generado por la metodología de desarrollo XP. Además, se define la arquitectura a utilizar y los patrones de diseño, así como la estimación del esfuerzo por cada una de las HU identificadas, como punto de partida para elaborar el plan de iteraciones y obtener el plan de entrega final.

2.1 Propuesta del sistema

Este trabajo propone una solución a los problemas encontrados en las herramientas existentes en el mercado, según los parámetros analizados. La aplicación, llamada **MonitorQoS**, brinda un entorno simple e intuitivo para el análisis de parámetros de calidad de servicio, el cual puede ser usado por usuarios sin mucha experiencia y sin necesidad de configuraciones tediosas.

En la figura 4 se muestra un diagrama del despliegue de la aplicación:

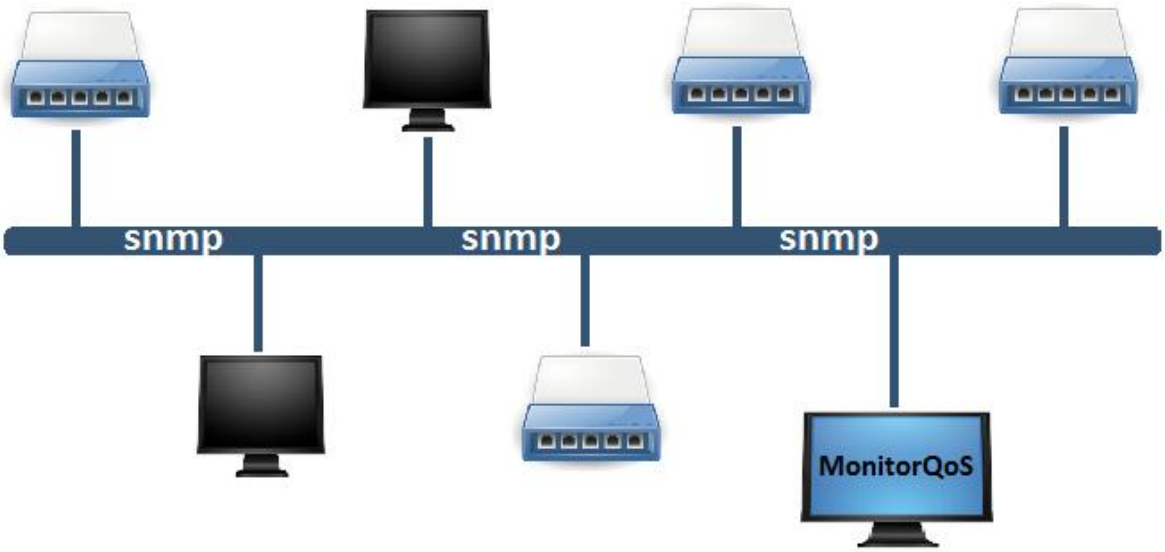

Figura 4: Arquitectura de despliegue.

En la figura se describe el despliegue de la aplicación MonitorQoS en un ambiente real. La aplicación se instalará en una PC administradora, y será capaz de interactuar con los diferentes dispositivos de la red, los cuales deben tener un sistema agente que permita comunicarse con la herramienta y brindarle información de las variables de la MIB ubicada en cada dispositivo.

La solución presentada para el desarrollo de la herramienta realiza mediciones de los parámetros de QoS: latencia, caudal, pérdida de paquetes y disponibilidad. La aplicación podrá ser desplegada en cualquier estación que cumpla con los requisitos mínimos establecidos y será capaz de monitorear cualquier equipo en la red que posea un agente o con soporte para la versión v2c.

Cumpliendo con las fases de desarrollo de XP, se desarrollan a continuación un conjunto de pasos los cuales se describen en los siguientes epígrafes.

2.1.1 Roles del sistema

Los roles constituyen las responsabilidades que se le atribuyen a los involucrados con el sistema, los mismos representan a los actores que interactúan con la aplicación y que obtienen un resultado de valor a raíz de una interacción determinada entre actor-sistema.

Roles	Descripción

Cliente	Hace uso de la herramienta, beneficiándose de sus funcionalidades; es el encargado de realizar los análisis pertinentes, a partir de la información brindada por la aplicación.

2.2 Requisitos de la solución

Según la IEEE[10], los requisitos para un sistema de software determinan las condiciones o capacidades que debe exhibir o poseer el mismo para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta. Los requisitos de desarrollo se clasifican en **funcionales** y **no funcionales**.

2.2.1 Requisitos Funcionales

Los requisitos funcionales (RF) definen una función del sistema o sus componentes, estos establecen los comportamientos del sistema. En el epígrafe 2.3 se describen los requisitos en las Historias de Usuarios.

- RF1. Insertar dispositivo para su monitorización.
- RF2. Modificar los parámetros de conexión de un dispositivo.
- RF3. Eliminar un dispositivo.
- RF4. Buscar un dispositivo.
- RF5. Mostrar el valor correspondiente a la **latencia** calculada de cada dispositivo.
- RF6. Mostrar el valor correspondiente al **caudal** calculado de cada dispositivo.
- RF7. Mostrar el valor correspondiente a la **pérdida de paquetes** calculada de cada dispositivo.
- RF8. Mostrar el valor correspondiente a la **disponibilidad** calculada de cada dispositivo.

*[10] IEEE: **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers o Instituto de Ingenieros Eléctricos y Electrónicos es una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.*

- RF9. Crear una base de datos histórica.
- RF10. Elaborar reporte de los valores de cada parámetro de QoS en un rango de fechas.
- RF11. Elaborar gráfica de los valores de cada parámetro de QoS en un rango de fechas.

2.2.2 Requisitos No Funcionales

Un requisito no funcional (RNF) o atributo de calidad es un requisito que define criterios que pueden usarse para medir y evaluar la operación de un sistema; son cualidades de rendimiento, calidad, seguridad, etc. que deben cumplirse para que el sistema tenga un desempeño aceptable para el cliente. Los RNF no se describen en las Historias de Usuario ya que son de aspecto técnico, y un tanto difícil de entender para los usuarios sin experiencia.

Usabilidad

RNF1.1. Brindar una interfaz amigable, sencilla e intuitiva, de manera que cualquier usuario sin experiencia pueda usarlo y se reduzcan los tiempos de entrenamiento, soporte y prueba por parte de los mismos.

Eficiencia

RNF2.1. Capacidad para monitorizar una gran cantidad de dispositivos de manera eficiente.

RNF2.2. Brindar tiempos de respuestas aceptables, por lo que se establece un umbral de tiempo límite máximo (1 minuto) en el que se deben mostrar los parámetros encuestados.

RNF2.3. Aprovechamiento al máximo de los recursos de hardware sobre el que se está ejecutando la herramienta.

Software

RNF3.1. Se debe contar con una plataforma GNU/Linux o Windows, con las bibliotecas ***rrdtool.1.4*** o superior instaladas.

Hardware

RNF3.2. Para el correcto funcionamiento de la aplicación se debe disponer de una computadora con microprocesador *Core 2 Duo* o superior, con 1 GB de RAM o más, 50 MB de espacio libre en disco duro o superior.

2.3 Historias de Usuario (HU)

El ciclo de vida de un proyecto según la metodología XP se inicia con la fase de exploración. En esta fase los clientes plantean de manera general las HU que son de interés para la entrega del producto. A la par, el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que serán usadas para desarrollar la solución. Se prueba la tecnología y se define la arquitectura del sistema construyendo prototipos.

Las **HU** son tarjetas de papel en las cuales el cliente describe, en un lenguaje natural, cómo desea que el sistema se comporte. Las descripciones son cortas y sencillas, aunque los desarrolladores pueden aportar ideas para ayudar al cliente a identificar sus necesidades.

El tratamiento de las HU es muy dinámico y flexible. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarlas en unas semanas. [Letelier and Penadés 2010]

Fase 1: Exploración:

En esta fase, los clientes plantean a grandes rasgos las HU que son de interés para la primera entrega del producto, las cuales representan una breve descripción del comportamiento de la aplicación. A continuación se describen 4 de las 11 historias de usuarios, las restantes podrán ser consultadas en el documento “Historias de Usuario”.

Historia de Usuario	
Número: 1	**Nombre:** Insertar dispositivo
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Nivel de Complejidad:** Alta
Programador Responsable: José Arias Pérez	
Descripción: El sistema debe permitir insertar dispositivos para su monitorización. Para insertar un dispositivo, se debe seleccionar la opción ***Insertar***, luego se solicitan los parámetros para la conexión con el dispositivo: dirección IP (es lo que identifica al dispositivo dentro de la red), la comunidad de acceso (se define como la contraseña para poder acceder a las variables que	

brinda SNMP), el período de encuesta (es el tiempo que transcurre entre una encuesta y otra a los dispositivos), así como una descripción que desee darle el usuario. Luego de añadido el dispositivo, debe mostrarse en la interfaz principal junto con los valores de las variables de calidad de servicio de cada una de las interfaces que posee el dispositivo.
Observaciones: Se deben validar los datos de entrada para verificar que cumplan con los requisitos sintácticos de cada parámetro. (Ej. La dirección IP del dispositivo debe estar compuesta por cuatro octetos.)

Historia de Usuario	
Número: 2	**Nombre:** Modificar dispositivo
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Nivel de Complejidad:** Media
Programador Responsable: José Arias Pérez	
Descripción: El sistema debe permitir modificar los parámetros de conexión de los dispositivos existentes. Para ello se debe seleccionar el dispositivo de la lista, y seleccionar la opción ***Modificar***, y luego se muestran los parámetros de conexión del dispositivo seleccionado. Posteriormente el usuario puede establecer los nuevos parámetros para el dispositivo.	
Observaciones: Se deben validar los datos de entrada para verificar que cumplan con los requisitos sintácticos de cada parámetro. (Ej. La dirección IP del dispositivo debe estar compuesta por cuatro octetos.) Las bases de datos RRD, utilizadas para almacenar los datos recolectados de los dispositivos, son sensibles al tiempo, por lo que una modificación al dispositivo provocará la reiniciación de las mismas y por tanto se perderán los datos históricos almacenados hasta el momento.	

Historia de Usuario	
Número: 3	**Nombre:** Eliminar dispositivo
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Nivel de Complejidad:** Media
Programador Responsable: José Arias Pérez	
Descripción: El sistema debe permitir eliminar un dispositivo del listado de dispositivos existentes. Para ello se debe seleccionar el dispositivo de la lista, y seleccionar la opción ***Eliminar***, luego el dispositivo debe ser eliminado de la lista y cancelar las peticiones al mismo.	
Observaciones:	

Historia de Usuario	
Número: 5	**Nombre:** Mostrar latencia calculada
Actor: Usuario	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Nivel de Complejidad:** Alta
Programador Responsable: José Arias Pérez	
Descripción: El sistema debe mostrar los valores correspondientes a la latencia calculada desde la estación administradora hacia el dispositivo. Este valor es calculado en cada intervalo de tiempo (período de encuesta) establecido por el usuario al insertar el dispositivo, y se muestra en la interfaz de la aplicación.	

Observaciones:

Fase 2: Planificación de la entrega

En esta fase el cliente establece la prioridad de cada HU, y posteriormente los programadores realizan una estimación del esfuerzo necesario para la realización de cada una de ellas.

Plan de duración de iteraciones

Aquí se definen las HU que serán implementadas en cada iteración del proceso de desarrollo, así como las fechas en que serán liberadas. El proceso de desarrollo se dividió en tres etapas o iteraciones, y a continuación se muestran las HU y la iteración a la que pertenece cada una.

No. de Iteración	**Historia de Usuario**	**Estimación de duración total (*semanas*)**
1	Insertar un dispositivo	**5**
	Modificar un dispositivo	
	Eliminar un dispositivo	
	Buscar dispositivo	
2	Mostrar **caudal** calculado de los dispositivos.	**8**
	Mostrar **pérdida de paquetes** calculada de los dispositivos	
	Mostrar **disponibilidad** calculada de los dispositivos.	

	Mostrar **latencia** calculada de los dispositivos	
3	Crear base de datos histórica	**5**
	Elaborar **reporte** de los valores de cada parámetro de QoS en un rango de fechas.	
	Elaborar **gráfica** de los valores de cada parámetro de QoS en un rango de fechas.	
Duración Total		**18**

Plan de entrega

El plan de entrega es un artefacto generado en la fase de planificación que permite realizar una organización del tiempo de las entregas. Esto sirve como guía para el desarrollo en tiempo de la aplicación en cuestión.

Artefacto	***Iteración***	***Fecha Liberación***
MonitorQoS v1.0	Final de la primera iteración.	15 al 20 de febrero.
MonitorQoS v1.1	Final de la segunda iteración.	1 al 5 de abril.
MonitorQoS v1.3 Final	Final de la tercera iteración.	1 al 15 de mayo.

2.4 Diseño del sistema

Luego de realizada la planificación de las actividades se procede a la fase de diseño del sistema. En esta fase se modelan los diagramas que servirán de guía en el proceso de desarrollo. También se obtienen las tarjetas CRC donde se ilustran las clases y la información necesaria a almacenar en cada una de ellas.

2.4.1 Tarjetas CRC del sistema

Una tarjeta CRC es una técnica para la representación de sistemas orientados a objetos. Son utilizadas para representar las responsabilidades de las clases y sus interacciones. Sus siglas representan los conceptos de **Clase**, **Responsabilidades** y **Colaboración**.

Una clase define cualquier objeto que se quiera representar. Las responsabilidades de una clase son las cosas que la definen y las tareas que realizan: sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades, es decir sus relaciones con otras clases.

A continuación se describen las tarjetas CRC más relevantes, las restantes podrán ser consultadas en el documento “Tarjetas CRC”.

Tarjeta CRC 1	
Clase: *SNMPTransport*	
Responsabilidades:	**Colaboraciones:**
Es la encargada de manejar la comunicación con los dispositivos. Almacena todos los atributos necesarios para la conexión a nivel del protocolo SNMP; contiene las estructuras necesarias para trabajar con la biblioteca Net-SNMP que, a su vez, realiza la comunicación con el dispositivo y devuelve los datos mediante funciones 'callback'. ***createSession*** *(…);* ***closeSession*** *();* ***asyncSnmpGet*** *(…);* ***asyncSnmpGetIfTable*** *(…);* ***startBulkRequests*** *();* ***handleSnmpReq*** *(…);* ***handleSnmpRes*** *(…);* ***handleTimeout*** *(…);* ***responseCallBack*** *(…);* ***readResponseCallBack*** *(…);*	

Tarjeta CRC 2	
Clase: *Grapher*	
Responsabilidades:	**Colaboraciones:**
Es la encarga de las operaciones sobre la base de datos RRD. Contiene las consultas para la creación de la base de datos, su actualización, creación de reportes y gráficos a través del tiempo. ***createDatabases*** *();* ***updateInterface*** *(…);* ***graphBandwidthUseMbps*** *(…);* ***graphBandwidthUsePerc*** *(…);* ***graphPacketLoss*** *(…);* ***reportBandwidthUseMbps*** *(…);* ***reportBandwidthUsePerc*** *(…);* ***reportPacketLoss*** *(…);*	

Tarjeta CRC 3	
Clase: *Device*	
Responsabilidades:	**Colaboraciones:**
Representa los objetos de los dispositivos; engloba las propiedades de los dispositivos, los atributos necesarios para la recolección y salva de los datos en la base de datos. Es la encargada de planificar las tareas de lectura en los dispositivos para la recolección de los datos.	*SNMPTransport* *ICMPPinger* *Grapher*

connect *();* ***reconnect*** *();* ***addReadTask*** *();* ***deviceRead*** *();* ***updateInterfaceGraph*** *(…);* ***graph*** *(…);* ***report*** *(…);*	

Tarjeta CRC 4	
Clase: *Collector*	
Responsabilidades:	**Colaboraciones:**
Es la encargada de la creación de los dispositivos, su modificación y eliminación. Se encarga también del manejo de los hilos que usará la aplicación para optimizar las operaciones de encuesta a los dispositivos. Es la clase que controla todas las operaciones en la lógica de negocio. ***startCollector*** *();* ***addDevice*** *(…);* ***modifyDevice*** *(…);* ***deleteDevice*** *(…);* ***graphDevice*** *(…);* ***reportDevice*** *(…);*	*Device* *ThreadPool*

2.4.2 Patrones de diseño

Un patrón es una solución reutilizable de problemas recurrentes que ocurren durante el desarrollo del software, que ayudan a entender las soluciones del problema con un vocabulario igual, lo que permite un mejor entendimiento.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Características de los patrones de diseño:

- Son soluciones concretas y técnicas.
- Se utilizan en situaciones frecuentes.
- Favorecen la reutilización de código. [Hernández 2010]

Para el desarrollo de la aplicación se tuvo en cuenta el empleo de patrones GRASP, los cuales brindan mayor robustez al sistema a la hora de su desarrollo.

Patrones GRASP[11] utilizados:

- ***Controlador:*** Es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto es para aumentar la reutilización de código y a la vez tener un mayor control. [Villalobos 2012]

  Este patrón se observa en la clase ***DeviceItem*** (ver figura 7) que es la encargada de recibir los datos de las consultas realizadas a los dispositivos, así como todos los eventos relacionados a los mismos, y luego enviarlos a la clase encargada de mostrarlos en la interfaz al usuario.

- ***Creador:*** El patrón creador ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización. [Villalobos 2012]

  Se evidencia en la clase **SimpleNetworkMonitor** (ver figura 7) que es la encargada de crear, modificar y eliminar los objetos **Device**, los cuales representan a los dispositivos en el sistema.

[11] GRASP: **G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns o Patrones Generales de Software para Asignación de Responsabilidades, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software.

- **Alta cohesión:** Este patrón propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables, asumiendo solamente las responsabilidades que deben manejar cada clase, evadiendo un trabajo excesivo. Su utilización mejora la claridad y facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras de funcionalidad, generan **bajo acoplamiento**, soporta mayor capacidad de reutilización. [Leon 2011]

  La alta cohesión se puede evidenciar con las clases ***SNMPTransport*** e ***ICMPPinger*** (ver figura 7), las cuales separan las responsabilidades de comunicación con los dispositivos, a la vez que colaboran juntas en la clase ***Device*** para obtener los parámetros necesarios.

2.4.3 Estándares de codificación

CamelCase o **Medial Capitals** es una notación para escribir palabras o frases, de tal manera, que formen una sola palabra juntas, con las letras iniciales de cada palabra en mayúsculas. Esta notación puede comenzar con letra inicial mayúscula o minúscula.

Esta es la notación usada en el desarrollo de la solución, siguiendo las siguientes reglas:

- Los nombres de clases comienzan con letra mayúscula, siguiendo la estructura CamelCase. (ej. *UnaClase*)

- Los nombres de variables o constantes de las clases, comienzan con letra minúscula, y terminan con _. (ej. *unaVariable_*)

- Los nombres de variables o constantes auxiliares y los nombres de funciones, serán con letra minúscula solamente. (ej. *unaVariable, unaFuncion()*)

A continuación se muestra, en la figura 5, un ejemplo de la codificación descrita anteriormente.

```
11
12  class Grapher
13  {
14
15  public:
16      Grapher(string databasePath, unsigned int period);
17      virtual ~Grapher();
18
19      //static string epochToSecs(posix_time::ptime& time_);
20      void update( ResponseMap& result );
21
22      void reportTraffic( time_t, time_t );
23
24      void graphTraffic( time_t, time_t );
25      void graphBandwidthUse( time_t, time_t );
26      void graphPacketLoss( time_t, time_t );
27
28      // Getters and Setters
29      string getDatabasePath() { return databasePath_; }
30      time_t getStartTime();
31      time_t getNow();
32
33  private:
34      void prepareTrafficGraph();
35      void prepareBandwidthGraph();
36      void preparePacketLossGraph();
37
38      time_t startTime_;
39      string databasePath_;
40      unsigned long period_;
41
```

Figura 5: Ejemplo de la codificación usada en la herramienta.

En la figura 5 se observa el uso del estándar de codificación descrito anteriormente.

2.4.4 Patrón de arquitectura

El patrón de arquitectura **MVC**[12] permite realizar la programación multicapa, separando en tres componentes distintos los datos de una aplicación, la interfaz del usuario y la lógica de control. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

En MVC, el **Modelo** representa los datos que se envían y reciben para su visualización: cada tipo de información tiene su propio modelo. La **Vista** presenta la información a los usuarios. El

[12] MVC: **M**odel **V**iew **C**ontroller o Modelo Vista Controlador.

Controlador actúa de mediador entre la vista y el usuario, convirtiendo las acciones del usuario en peticiones para navegar por la aplicación, las cuales la vista transmite al modelo según sea necesario. Qt provee una arquitectura **Modelo/Vista** inspirada en el MVC. En Qt, en lugar de un controlador, se usa una abstracción ligeramente diferente: los delegados. Estos son los encargados de mostrar y manejar los componentes visuales de Qt necesarios para la interacción de los usuarios con la aplicación. [Blanchette and Summerfield 2006]

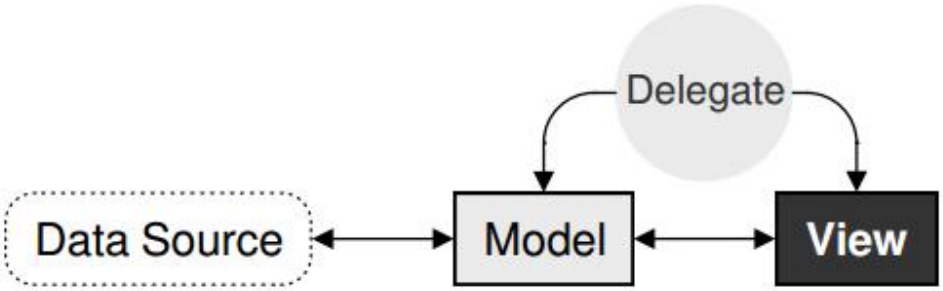

Figura 6: Arquitectura Modelo/Vista de Qt.

2.4.5 Diagrama de Clases del Diseño

A continuación se presenta en la figura 7 el diagrama de clases del diseño, en el cual se refleja la separación, en cada capa, de las diferentes clases del sistema.

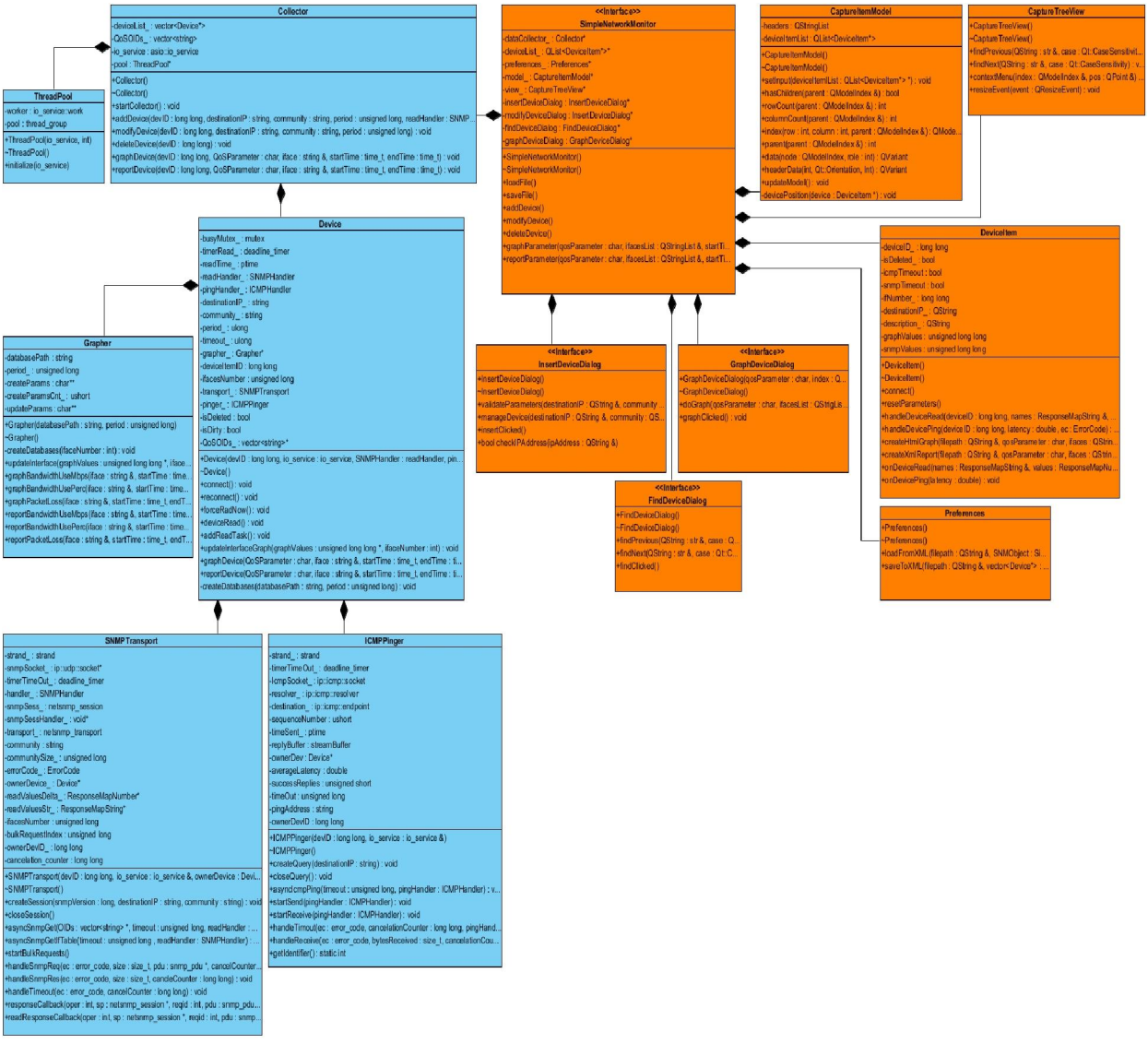

Figura 7: Diagrama de clases del diseño.

2.5 Conclusiones parciales

La metodología de desarrollo seleccionada para el trabajo posibilitó entre otras cosas que:

- Se definieran junto con el cliente las historias de usuarios, 11 en total, en un lenguaje natural y entendible para el mismo, facilitando el trabajo a la hora de implementar las funcionalidades.

- Se realizara la estimación del esfuerzo que conlleva la implementación de cada funcionalidad, manteniendo el margen de tiempo establecido.

- Se especificara el diseño de clases del sistema mediante la utilización de las tarjetas CRC, 14 en total, las cuales describen el sistema implementado y sus relaciones de clases.

- Además, se definieran los patrones de diseño a utilizar en la fase de desarrollo, haciendo uso de los patrones GRASP Controlador, Creador y Alta Cohesión, para aumentar la robustez del sistema.

Luego de este análisis se procede a la implementación del sistema, guiado por todo lo anteriormente establecido, lo que dará paso, posteriormente a la validación del sistema, mediante distintos tipos de pruebas.

CAPÍTULO 3: VALIDACIÓN DEL SISTEMA PROPUESTO

Introducción

En el presente capítulo se procede a la fase de validación del sistema implementado a lo largo de la fase de desarrollo. Se verifican si todas las historias de usuario fueron implementadas correctamente y cumpliendo todos los requisitos planteados por el cliente. Posteriormente se procede a realizar un conjunto de pruebas de aceptación y unitarias para validar la solución, además se agregan pruebas para la validación de la correcta medición, a partir de la comparación con otras herramientas.

3.1 Pruebas

Uno de los pilares de la Programación Extrema (XP) es el proceso de pruebas. Las pruebas son procesos que permiten verificar y revelar la calidad de un producto. Son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un software. XP divide las pruebas del componente en dos grupos: **pruebas unitarias**, encargadas de verificar el código y diseñada por los programadores, y **pruebas de aceptación** o pruebas funcionales, destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente. [Gutiérrez et al. 2010]

3.1.1 Pruebas unitarias

Las pruebas unitarias permiten probar el código de un componente para asegurar el correcto funcionamiento de cada uno por separado y luego se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión. El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas.

La realización de pruebas unitarias al sistema brindó una serie de ventajas al proceso de implementación del sistema:

- **Código documentado:** La documentación del código es una práctica muy útil a la hora de hacer chequeos y buscar fallos; brinda legibilidad y limpieza al código y ayuda a la hora de realizar mantenimiento al sistema.

- **Errores acotados y fáciles de localizar:** Las pruebas unitarias ayudan a detectarlos y corregirlos.

- **Fomentan el cambio:** Las pruebas unitarias facilitan al programador a que cambie el código para mejorar su estructura y diseño. Además, permiten hacer pruebas sobre los nuevos cambios y así asegurarse de que estos no introducen nuevos errores.

Una de las pruebas unitarias realizadas fue el **camino básico**, el cual permite obtener una medida de la complejidad de un diseño procedimental y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución, diseñando casos de prueba que garanticen que cada camino se ejecuta al menos una vez.

A partir de dicho análisis se puede obtener una medida cuantitativa de la dificultad de crear pruebas automáticas del código y también es una medición orientativa de la fiabilidad del mismo.

La complejidad ciclomática consta de tres formas para calcularse: [McCabe 1976]

V (G) = R, donde **R** es el número de regiones del grafo.

V (G) = A - N + 2, donde **A** es el número de arcos del grafo y **N**, el número de nodos.

V (G) = P + 1, donde **P** es el número de regiones rodeadas completamente por arcos del grafo.

A continuación se muestra la figura 8, con el código del método ***deleteDevice ()***, al cual se le realizará la prueba unitaria:

```
236 void SimpleNetworkMonitor::deleteDevice()
237 {
238     if( view->currentIndex().isValid() )
239     {
240         int pos = proxyModel->mapToSource(view->currentIndex()).row();
241
242         if( pos >= 0 && pos < deviceItemList.size() )
243         {
244             DeviceItem* dev = deviceItemList.takeAt( pos );
245
246             // The collector deletes Device and DeviceItem in the next request
247             dataCollector->deleteDevice( dev->getDeviceID() );
248
249             model->updateModel();
250         }
251     }
252 }
```

Figura 8: Código del método deleteDevice ().

A continuación se muestran los pasos requeridos para realizar la prueba de camino básico al código anterior:

Paso 1: *Generar el grafo correspondiente al código del método: [ver figura 9]*

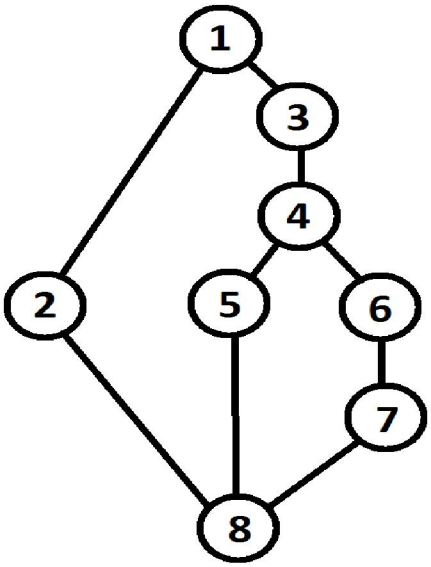

Figura 9: Grafo del método deleteDevice ().

Paso 2: *Se definen las áreas o regiones en que se divide el grafo generado: [ver figura 10]*

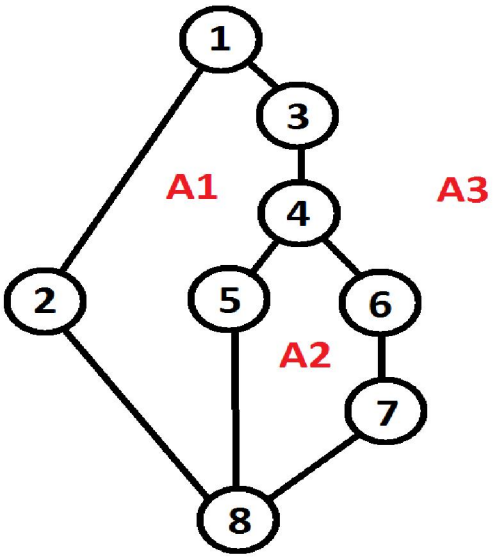

Figura 10: Áreas divididas en el grafo.

Paso 3: *Se procede a calcular la complejidad ciclomática del método:*

V (G) = 3

V (G) = 9 - 8 + 2 = 3

V (G) = 2 + 1 = 3

Una vez calculada la complejidad ciclomática del fragmento de código, se puede determinar el riesgo que supone utilizando los siguientes rangos: [McCabe 1976]

Complejidad Ciclomática	***Evaluación del Riesgo***
1 - 10	Programa simple, sin mucho riesgo.
11 - 20	Más complejo, riesgo moderado.
21 - 50	Complejo, programa de alto riesgo.
50 o más	Programa no comprobable, muy alto riesgo.

Ahora se puede determinar que existen 3 caminos linealmente independientes, lo cual, según la tabla anterior, determina que es un método simple y sin mucho riesgo. La complejidad ciclomática indica el número exacto de casos de prueba necesarios para probar cada punto de decisión en un programa. Por tanto, este método requiere de al menos tres casos de prueba para probar correctamente su validez.

3.1.2 Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra que se realizan a partir de las HU. Durante una iteración la HU seleccionada se convertirá en una prueba de aceptación. [Allende 2006]

Una HU puede tener todas las pruebas de aceptación que desee para asegurar su funcionamiento. El objetivo específico de esta prueba es garantizar que los requerimientos han sido cumplidos y que el sistema ha sido aceptable. [Beck 2000]

Como resultado de las pruebas de aceptación se obtendrán artefactos descritos en tablas llamadas **Caso de Prueba de Aceptación**.

Caso de Prueba de Aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Insertar un dispositivo.	
Descripción: Prueba para la funcionalidad de insertar un nuevo dispositivo en el sistema.	
Condiciones de ejecución: El usuario debe hacer clic en el botón ***Insertar dispositivo*** y se le muestra una interfaz para introducir los datos de entrada.	
Entradas de ejecución: • Se especifica un IP válido del dispositivo que se quiere monitorizar. • Se introduce la comunidad snmp del dispositivo, la cual debe conocerse de antemano. • Se introduce el período de encuesta deseado para el dispositivo. • Se introduce una descripción opcional para identificar al dispositivo.	
Resultado esperado: El dispositivo se añade sin errores y se muestra en la interfaz principal, de lo contrario, señala los campos que contienen errores en la entrada.	
Evaluación de la prueba: Prueba satisfactoria	

Caso de Prueba de Aceptación	
Código: HU2_P1	**Historia de usuario:** 2
Nombre: Modificar un dispositivo.	
Descripción: Prueba para la funcionalidad de modificar un dispositivo en el sistema.	
Condiciones de ejecución: El usuario debe seleccionar un dispositivo previamente añadido en la interfaz, a continuación hacer clic en el botón ***Modificar dispositivo*** o clic derecho y seleccionar dicha opción, luego se muestra una interfaz con los datos del dispositivo seleccionado.	

Entradas de ejecución: • Se procede a cambiar los datos deseados introduciendo los nuevos.
Resultado esperado: El dispositivo se modifica sin errores, de lo contrario, señala los campos que contienen errores en la entrada.
Evaluación de la prueba: Prueba insatisfactoria. Los datos no se validaron correctamente, por lo que se permitían dispositivos con campos obligatorios vacíos.

Caso de Prueba de Aceptación	
Código: HU2_P2	**Historia de usuario:** 2
Nombre: Modificar un dispositivo.	
Descripción: Prueba para la funcionalidad de modificar un dispositivo en el sistema.	
Condiciones de ejecución: El usuario debe seleccionar un dispositivo previamente añadido en la interfaz, a continuación hacer clic en el botón ***Modificar dispositivo*** o clic derecho y seleccionar dicha opción, luego se muestra una interfaz con los datos del dispositivo seleccionado.	
Entradas de ejecución: • Se procede a cambiar los datos deseados introduciendo los nuevos.	
Resultado esperado: El dispositivo se modifica sin errores, de lo contrario, señala los campos que contienen errores en la entrada.	
Evaluación de la prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de usuario:** 3
Nombre: Eliminar un dispositivo.	
Descripción: Prueba para la funcionalidad de eliminar un dispositivo en el sistema.	
Condiciones de ejecución: El usuario debe seleccionar un dispositivo previamente añadido en la interfaz, a continuación hacer clic en el botón ***Eliminar dispositivo*** o clic derecho y seleccionar dicha opción, luego el dispositivo debe desaparecer de la interfaz principal.	
Entradas de ejecución:	
Resultado esperado: El dispositivo se elimina sin errores y se cancelan las peticiones periódicas al mismo.	
Evaluación de la prueba: Prueba insatisfactoria. En ocasiones, al eliminar dispositivos que estaban realizando operaciones en ese instante, la aplicación terminaba de forma inesperada.	

Caso de Prueba de Aceptación	
Código: HU4_P1	**Historia de usuario:** 4
Nombre: Buscar un dispositivo.	
Descripción: Prueba para la funcionalidad de buscar un dispositivo en el sistema.	
Condiciones de ejecución: El usuario debe seleccionar la opción buscar, luego se debe insertar un patrón de búsqueda para localizar los dispositivos que se desean.	
Entradas de ejecución: • Se debe especificar un patrón de búsqueda.	
Resultado esperado: Se señalan en la interfaz principal, uno a uno, los dispositivos que coinciden con el criterio de búsqueda especificado.	

Evaluación de la prueba: Prueba satisfactoria.

Caso de Prueba de Aceptación	
Código: HU10_P1	**Historia de usuario:** 10
Nombre: Elaborar reporte en una rango de fechas.	
Descripción: Prueba para la funcionalidad de elaborar un reporte que represente el comportamiento de un parámetro dado en el dispositivo, en un rango de fechas dado.	
Condiciones de ejecución: El usuario debe seleccionar un dispositivo previamente añadido en la interfaz, a continuación hacer clic derecho y seleccionar la opción ***Reportes***, luego en el submenú mostrado seleccionar el parámetro para el que se desea el reporte.	
Entradas de ejecución: • Primeramente se seleccionan las fechas inicial y final para el reporte. • Luego el parámetro QoS, si se desea cambiar. • Luego se seleccionan las interfaces del dispositivo, de las cuales se desea saber el comportamiento histórico. • Luego se debe seleccionar la ruta en que se quiere guardar el archivo.	
Resultado esperado: Se debe generar un archivo XML, el cual contiene los valores calculados a través del tiempo en las interfaces seleccionadas y el rango de fechas especificado.	
Evaluación de la prueba: Prueba insatisfactoria. El archivo generado no podía ser visualizado correctamente ya que, debido a que la aplicación necesita permisos administrativos para ejecutarse, los ficheros generados se creaban con estos permisos también.	

Caso de Prueba de Aceptación	
Código: HU11_P1	**Historia de usuario:** 11
Nombre: Elaborar gráfica en una rango de fechas.	
Descripción: Prueba para la funcionalidad de elaborar una gráfica que represente el comportamiento de un parámetro dado en el dispositivo, en un rango de fechas dado.	
Condiciones de ejecución: El usuario debe seleccionar un dispositivo previamente añadido en la interfaz, a continuación hacer clic derecho y seleccionar la opción ***Gráficas***, luego en el submenú mostrado seleccionar el parámetro que desea graficar.	
Entradas de ejecución: • Primeramente se seleccionan las fechas inicial y final para la gráfica. • Luego el parámetro QoS, si se desea cambiar. • Luego se seleccionan las interfaces del dispositivo, de las cuales se desea saber el comportamiento histórico. • Luego se debe seleccionar la ruta en que se quiere guardar el archivo.	
Resultado esperado: Se debe generar un archivo de página web HTML, el cual contiene las gráficas del parámetro requerido, en las interfaces seleccionadas y el rango de fechas especificado.	
Evaluación de la prueba: Prueba satisfactoria	

Cada caso de prueba insatisfactorio genera una no conformidad, las cuales deben ser auto-explicables y relacionadas con el punto del sistema y deben ser tan concisas como sea posible. Las no conformidades fueron clasificadas según su complejidad de tres formas:

Tipo de complejidad de No Conformidades:

- Alta
- Media
- Baja

Las principales no conformidades encontradas fueron: errores ortográficos en las interfaces, errores de validación, mensajes de información mal elaborados, entre otros. Después de corregir los errores detectados en las pruebas, se pudo evidenciar que las mismas fueron satisfactorias. Se comprobó el correcto funcionamiento de la aplicación, cumpliéndose así con las funcionalidades descritas en las HU.

En la figura 11 se presentan los resultados de las no conformidades encontradas en cada iteración de pruebas, clasificadas por los tipos definidos; al final de la tercera iteración se concluyen con una aplicación funcional y libre de errores.

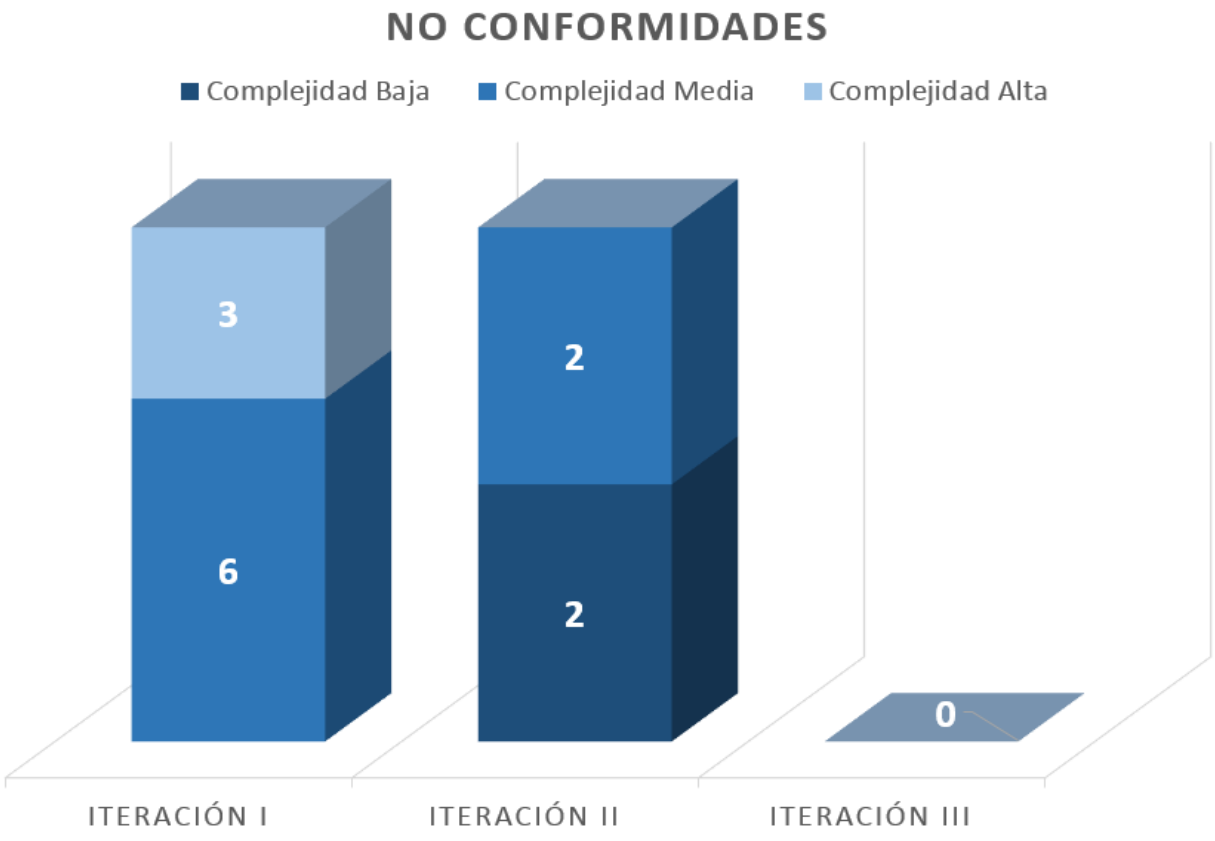

Figura 11: No conformidades detectadas en cada iteración de pruebas.

3.1.3 Validación de los resultados de las mediciones de la herramienta

Además de las pruebas unitarias y de aceptación realizadas, se realizaron pruebas de comparación con algunas herramientas profesionales. Las comparaciones realizadas tenían como objetivo probar la validez de las mediciones de la herramienta desarrollada.

Para estas pruebas se contó con el acceso hacia el dispositivo ***NN2-Produccion (S9300) (****Quidway S9303 Huawei Versatile Routing Plataform Software****)***, el cual es un dispositivo de enrutamiento de la Universidad y sobre el cual se tiene un monitoreo constante por parte del Nodo Central. Al tener acceso a este dispositivo y a las herramientas que monitoreaban al mismo, se procedió a monitorear el equipo con la herramienta desarrollada por un período de 24 horas, y luego a la comparación de los datos obtenidos.

La herramienta con la que se realizó la comparación fue ***Cacti***, la cual es utilizada para la medición del tráfico que circula por las interfaces de los dispositivos. Luego de 24 horas se obtuvieron las gráficas, de ***Cacti*** y de la herramienta ***MonitorQoS***. A continuación se muestran dos gráficas hechas cada herramienta en las que se puede observar las semejanzas en las mediciones realizadas.

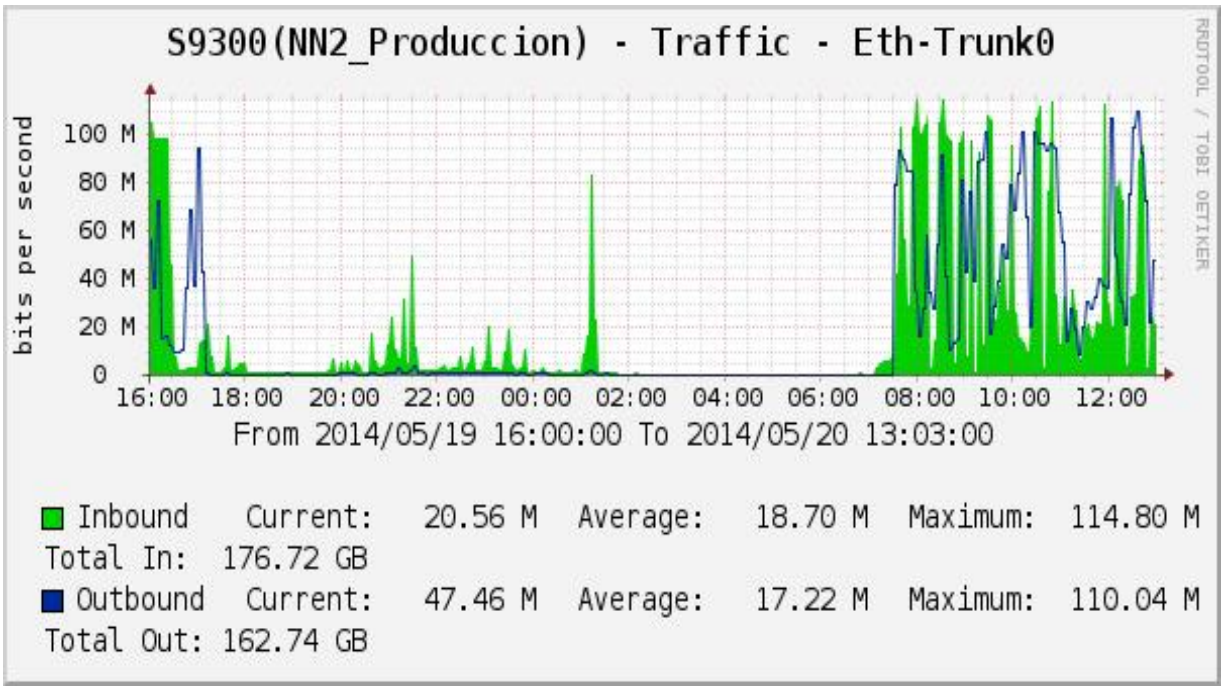

Figura 12: Gráfica de Cacti en un período de 24 horas

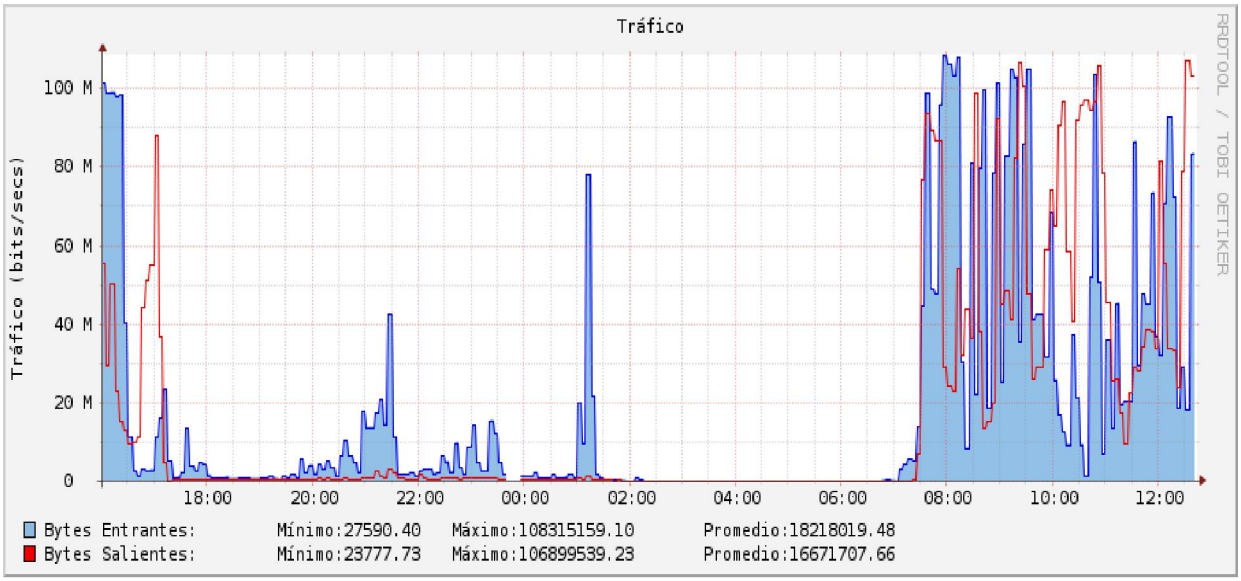

Figura 13: Gráfica de MonitorQoS en un período de 24 horas

Luego de las mediciones realizadas, se pudo observar las semejanzas de las mediciones en los mismos intervalos de tiempo, verificándose la validez para el parámetro caudal.

Otra de las pruebas realizadas fue para validar el parámetro latencia, para lo cual se realizaron varias encuestas al dispositivo anterior, usando la herramienta ***ping*** del sistema GNU-Linux Debian 6.0 *Squeeze* y la herramienta **MonitorQoS.** A continuación se muestran los resultados obtenidos en las siguientes figuras:

```
adrian@adrianPc:~$ ping 172.30.0.8
PING 172.30.0.8 (172.30.0.8) 56(84) bytes of data.
64 bytes from 172.30.0.8: icmp_req=1 ttl=252 time=4.16 ms
64 bytes from 172.30.0.8: icmp_req=2 ttl=252 time=12.2 ms
64 bytes from 172.30.0.8: icmp_req=3 ttl=252 time=4.28 ms
64 bytes from 172.30.0.8: icmp_req=4 ttl=252 time=6.10 ms
64 bytes from 172.30.0.8: icmp req=5 ttl=252 time=8.96 ms
64 bytes from 172.30.0.8: icmp_req=6 ttl=252 time=12.2 ms
64 bytes from 172.30.0.8: icmp_req=7 ttl=252 time=4.50 ms
64 bytes from 172.30.0.8: icmp_req=8 ttl=252 time=4.01 ms
64 bytes from 172.30.0.8: icmp_req=9 ttl=252 time=6.53 ms
64 bytes from 172.30.0.8: icmp_req=10 ttl=252 time=9.96 ms
```

Figura 14: Valores de la utilidad ping del sistema GNU-Linux Debian 6.0 Squeeze.

172.30.0.8	Huawei Router Fac5	5.00
172.30.0.8	Huawei Router Fac5	12.25
172.30.0.8	Huawei Router Fac5	4.50
172.30.0.8	Huawei Router Fac5	6.50
172.30.0.8	Huawei Router Fac5	8.25
172.30.0.8	Huawei Router Fac5	11.50
172.30.0.8	Huawei Router Fac5	4.50
172.30.0.8	Huawei Router Fac5	4.00
172.30.0.8	Huawei Router Fac5	6.33
172.30.0.8	Huawei Router Fac5	10.75

Figura 15: Valores de la herramienta MonitorQoS.

Al término de estas mediciones, se pudo observar las semejanzas de las mediciones para la latencia, con diferencias de pocos milisegundos, verificándose la validez para estas mediciones.

3.2 Conclusiones parciales

Las pruebas fueron realizadas en varias iteraciones, las cuales fueron arrojando poco a poco las deficiencias que presentaba la aplicación, posibilitando que se fueran corrigiendo los problemas de una iteración a otra. Se realizaron pruebas de aceptación, pruebas unitarias y pruebas de comparación para validar el sistema en todos los aspectos posibles. Las principales no conformidades encontradas fueron errores de interfaz y de validación, los cuales fueron corregidos a través de las iteraciones. Las pruebas realizadas en la tercera y última iteración arrojaron resultados positivos del sistema, el cual se mantuvo el sistema activo durante varios días, donde se comportó de manera estable y cumpliendo con las funcionalidades requeridas. Finalmente, se obtuvo una aplicación funcional sin errores.

CONCLUSIONES GENERALES

Durante la investigación se evidenció la necesidad de este trabajo, debido a las deficiencias de las herramientas actuales, las cuales no agrupan en su totalidad los parámetros de QoS requeridos, son difíciles de manejar y en ocasiones de licencia privativa.

El resultado final de este trabajo ha sido la obtención de la herramienta ***MonitorQoS*** para la medición de parámetros de calidad de servicio en una LAN. La misma cumple con los objetivos trazados al inicio de la investigación: realiza mediciones de latencia, caudal, disponibilidad y pérdida de paquetes, además que permite la creación de reportes y gráficas del comportamiento de los dispositivos a través del tiempo.

La solución fue desarrollada siguiendo una metodología ágil, específicamente XP, y utilizando una arquitectura MVC. Las HU que fueron solicitadas han sido cumplidas completamente. La fase de pruebas arrojó que el sistema se encuentra estable y completamente funcional; fueron realizadas pruebas unitarias y de aceptación para validar las interfaces de usuario y el código de las funcionalidades. El sistema fue validado mediante la comparación con otras herramientas profesionales: se compararon los parámetros caudal y latencia con respecto a la herramienta Cacti y al *PING* del sistema, arrojando resultados satisfactorios.

RECOMENDACIONES

Se recomienda el empleo de la herramienta para la medición de la QoS en la Universidad de las Ciencias Informáticas y en otras redes de área local.

REFERENCIAS BIBLIOGRÁFICAS

AGÉ, M., BAUDRU, S., CROCFER, N., CROCFER, R., EBEL, F., HENNECART, J., LASSON, S., PUCHE, D. AND RAULT, R. *Seguridad informática - Ethical Hacking: Conocer el ataque para una mejor defensa.* Edtion ed., 2011. ISBN 9782746068117.

ALBÁN, O.A.V. *Introducción al Qt y al Qt Creator*. Edtion ed. Universidad de Cauca, Colombia, 2008.

ALLENDE, R. *Desarrollo de Portales y Extranet con Plone*. Edtion ed., 2006.

ANDRADE, L.G.D. AND SEBASTIÁ, M.S. Evaluación de las prestaciones de una red EPON mediante el programa OPNET. 2005. Available from Internet:<http://w3.iec.csic.es/ursi/articulos_gandia_2005/articulos/TE1/6006.pdf>.

BECK *Planeando en Programación Extrema*. Edtion ed., 2000.

BLANCHETTE, J. AND SUMMERFIELD, M. *C ++ GUI Programming with Qt 4*. Edtion ed.: Prentice Hall, 2006.

CABLETRON SYSTEMS, I., BMC SOFTWARE, I. AND IBM *An Architecture for Describing SNMP Management Frameworks RFC 2571*. Edtion ed.: Network Working Group, 1999.

CASTILLO, O. AND SEVILLA, H. *Fases de la Programación Extrema.* Edtion ed., 2010.

CISCO SYSTEMS, I. How To Calculate Bandwidth Utilization Using SNMP. 2005. Available from Internet:<http://www.cisco.com/c/en/us/support/docs/ip/simple-network-management-protocol-snmp/8141-calculate-bandwidth-snmp.html>.

COMUNITY, N.-S. *Net-SNMP History*. In., 2011.

CRUZ, L.C. Sistema de Monitorización. In *Ingeniería Técnica Informática de Sistemas.* Universidad de Sevilla Universidad de Sevilla 2005.

CRUZ, M.D.R., MARTÍNEZ, R., VICIEDO, L.G. AND GARCIA, Y.S. Modelo analítico del protocolo 802.3 para la evaluación de la QoS. *Memorias del VI simposio Internacional de Telecomunicaciones en la XV Convención y Feria Internacional, Informática 2013* [Type of Work]. 2013. ISSN 978-959-7213-02-4.

DAWES, B. *Boost Background Information*. Edtion ed., 2011.

FERGUSON, P. AND HUSTON, G. *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*. Edtion ed.: John Wiley & Sons, 1998.

FLICKENGER, R., AICHELE, C.E., FONDA, C. AND FORSTER, J. *Wireless Networking in the Developing World*. Edtion ed. EUA, 2006.

GALLARDO, D. *Iniciándose en la plataforma Eclipse*. Edtion ed. Estados Unidos, 2012.

GROUP, N.W. *A Simple Network Management Protocol (SNMP) RFC 1157*. Edtion ed., 1990.

GUTIÉRREZ, J.J., ESCALONA, M.J. AND MEJÍAS, M. *Pruebas del sistema en Programación Extrema*. Edtion ed., 2010.

HERNÁNDEZ, P.V. *Uso de patrones de arquitectura*. Edtion ed., 2010.

ITU, U.I.D.T. Definiciones de los términos relativos a la calidad de servicio. 2008. Available from Internet:<https://www.itu.int/rec/T-REC-E.800-200809-I/es>.

LEON, J.G. Arquitectura y Diseño del Componente de Búsqueda de Indicadores de Perforación. (SEnDI). In. La Habana: Universidad de las Ciencias Informáticas, 2011.

LETELIER, P. AND PENADÉS, M.C. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP) 2010.

MCCABE, T.J. Complexity Measure. IEEE Transactions on Software Engineering, 1976.

MESA, J.L.M. Desarrollo de métodos de medición para evaluar la calidad de servicios en el acceso a internet. In *Facultad de Ingeniería Electrónica.* Perú: Universidad Nacional Mayor de San Marcos, 2003.

MICROSOFT, C. Fundamentos de Internet Control Message Protocol (ICMP). In., 2007.

OETIKER, T. MRTG The Multi Router Traffic Grapher. Proceedings of the Twelfth Systems Administration Conference, 1998.

OETIKER, T. RRDtool logging & graphing (http://oss.oetiker.ch/rrdtool/). In., 2013.

OVIEDO, G.E.A. AND INATTY, P.D.P. Herramienta para la gestión y administración de redes IPv4 e Pv6 mediante la utilización del protocolo SNMP In *Facultad de Ciencias, Laboratorio de Comunicación y Redes* Universidad Central de Venezuela, 2008.

PRADO, C. *Implantación de Calidad de Servicio (QOS) en Redes Inalámbricas WI-FI*. Edtion ed. Escuela Superior de Ingeniería Mecánica y Eléctrica. Culhuacan, México, D.F: Tesina, 2009.

ROCKWOOD, B. *The Net-SNMP Programming Guide*. Edtion ed., 2004.

SAYDAM, T. AND MAGEDANZ, T. From Networks and Network Management into Service and Service Management. Journal of Networks and Systems Management, 1996, vol. 4, no. 4.

SEVILLA, O.C. AND D.F, H. *Programación Extrema*. Edtion ed., 2010.

UIT-T. *Servicio de gestión de la red de gestión de las telecomunicaciones: Visión de conjunto.* In.: Unión Internacional de Telecomunicaciones, 1992, vol. 3200.

VILLALOBOS, J. Facultad de Ingeniería - Departamento de Ingeniería de Sistemas y Computación. In., 2012.

Y.1540, U.-T. *Aspectos del protocolo Internet – Calidad de servicio y características de red.* In., 2005.

BIBLIOGRAFÍA

1. Network Working Group, D. Harrington de Cabletron Systems Inc. R. Presuhn de BMC Software Inc. B. Wijnen de IBM T. J. Watson Research, *An architecture for describing SNMP management framework*, 2771, Abril 1999.
2. GROUP, N.W. *A Simple Network Management Protocol (SNMP)* RFC 1157, 1990.
3. OETIKER, T. RRDtool logging & graphing (*http://oss.oetiker.ch/rrdtool/*), 2013.
4. MRTG Official Page.[En línea] 2013, *http://www.mrtg.org/*
5. Cacti Official Page.[En línea] 2013, *http://www.cacti.net/*
6. PandoraFMS Official Page.[En línea] 2013, *http://www.pandorafms.org/*
7. What's Up GOLD Official Page.[En línea] 2013, *http://www.whatsupgold.com/*
8. OpenNMS Official Page.[En línea] 2013, *http://www.opennms.com/*
9. Pressman, R.S., *Ingeniería del Software. Un enfoque práctico.* 2012.
10. Sevilla, O.C.D.F.H. *Programación Extrema*. 2010; Available from: http://programacionextrema.tripod.com/index.htm.
11. Hernández, P.V., *Uso de patrones de arquitectura.* 2010.