UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

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

Título: "*Sistema de prueba de rendimiento para SoftPBX Asterisk*"

Autores:

Diovis Robinet Morales.

Yerandi Bracero Blanca.

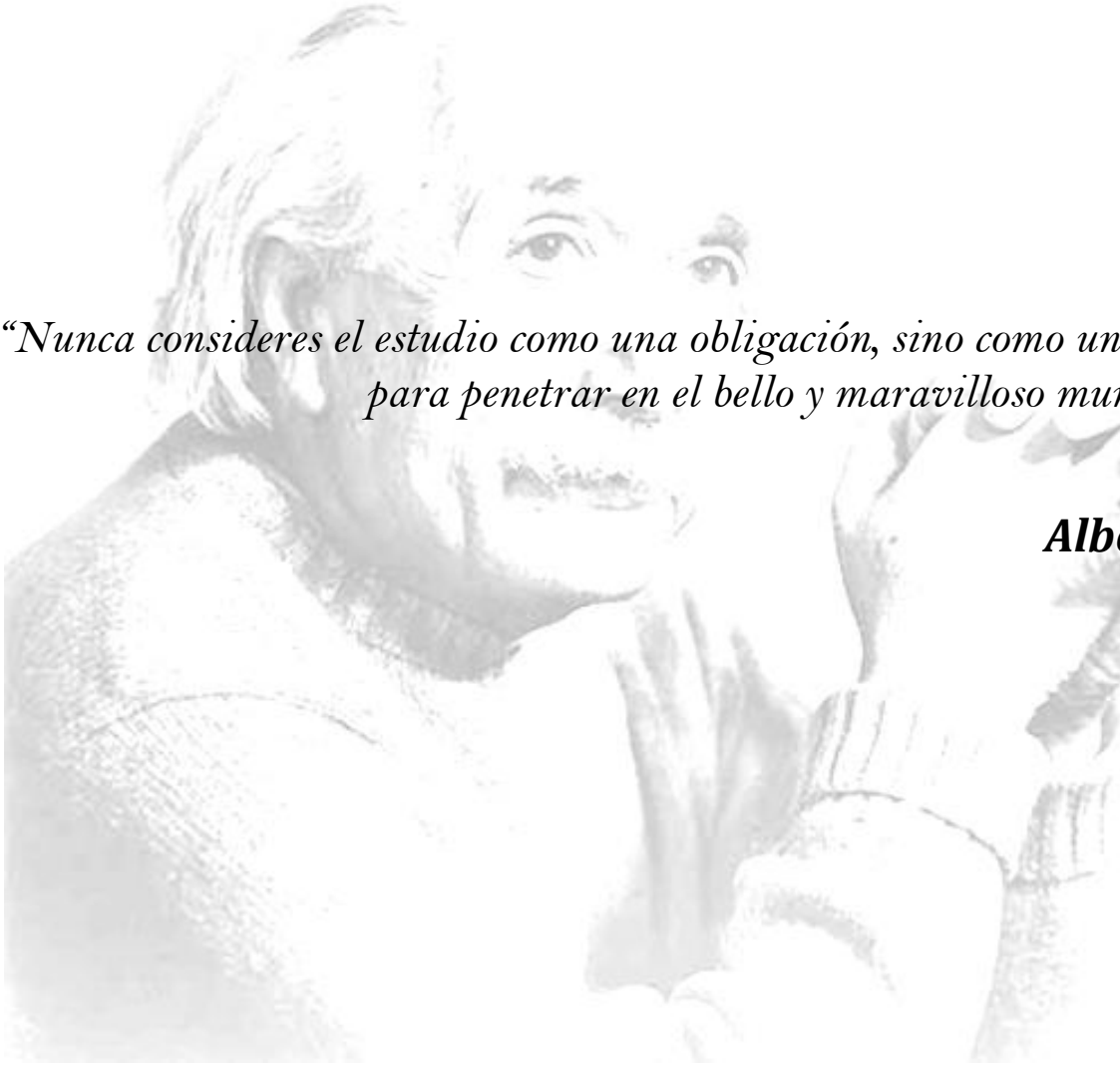
Tutores:

Ing. Darvis Dorvigny Dorvigny.

Ing. Iris Margarita Reina Heredia.

Ciudad de La Habana, Junio del 2009.

"Año del 50 Aniversario del Triunfo de la Revolución Cubana"



"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber".

Albert Einstein.

AGRADECIMIENTOS

La culminación del presente trabajo de tesis implicó mucho esfuerzo y dedicación, pero no hubiera sido posible sin el apoyo de todas las personas e instituciones que de una forma u otra han contribuido a la realización de este trabajo.

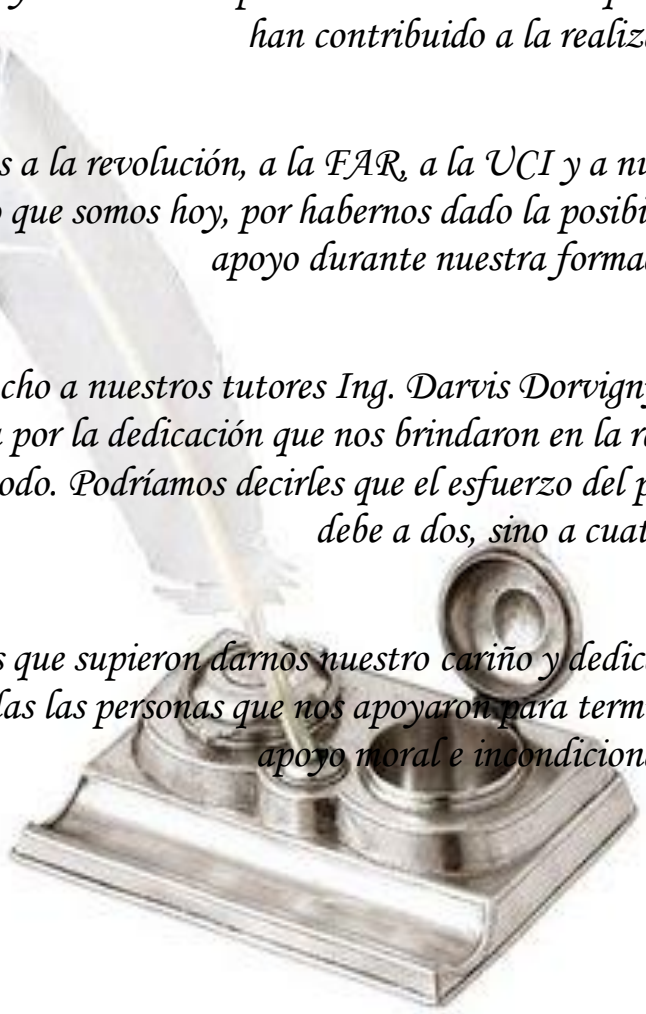
Le damos las gracias a la revolución, a la FAR, a la UCI y a nuestros profesores por habernos forjados de lo que somos hoy, por habernos dado la posibilidad de estudiar y el apoyo durante nuestra formación como ingenieros.

Les agradecemos mucho a nuestros tutores Ing. Darvis Dorvigny Dorvigny, Ing. Iris Margarita Reina Heredia por la dedicación que nos brindaron en la realización de nuestra tesis, muchas gracias por todo. Podríamos decirles que el esfuerzo del presente trabajo no se debe a dos, sino a cuatro, gracias a ustedes.

A todos nuestros amigos que supieron darnos nuestro cariño y dedicación, compañeros de aulas y de travesuras. A todas las personas que nos apoyaron para terminar el trabajo, por el apoyo moral e incondicional que nos brindaron.

Gracias a todos.

Los Autores.



DEDICATORIA

Le dedico mi tesis aquellas personas que sin duda las quiero mucho y han sido para mí la fuerza e inspiración de todas mis batallas dentro y fuera de aquí.

Para lo más grande que me ha dado la vida mi madre Adria Vivian Blanca Nodal por ser mi principal fuente de inspiración durante toda mi vida, por estar siempre conmigo en todo momento, sin ella realmente no se qué sería de mi. Gracias por ser mí guía y la luz de mi camino.

Al mejor de mis amigos, mi abuelo querido Giraldo Blanca Fonseca por saber ser en todos los momentos el mejor padre del mundo y lograr que me formara una persona de bien.

A mi tía Amarilis Blanca Nodal por si mi segunda madre "ama", gracias por tu apoyo, dedicación y por haberme acompañado en estos cinco años de mi carrera.

A mi hermanita Yairelis Amarilis Acosta Blanca, al final todo el esfuerzo que queda y pueda realizar estará encaminado en ti. Te quiero hermanita linda.

Serán siempre todos ustedes mi ejemplo a seguir, por su orientación y su lucha incansable.

Todos hacemos una familia unida.

Yerandi Bracero Blanca.

DEDICATORIA

Le dedico mi tesis a mi querida madre, a esa luna que me ha dado amor y sabiduría, a mi abuelita Tatica, a mis tíos Gelpis y Leonardito, a mi tía Nilsa, a mi primita Grettel, y a mi hermanita Yeni.

A todos muchas gracias por darme el apoyo que necesitaba cuando mami no estaba.

A mis hermanos de estudio por estar en los momentos difíciles, a Nora por su sensibilidad y cariño y a Aly por estar en el presente.

Diovis Robinet Morales.

DECLARACIÓN DE AUTORÍA.

Declaramos que somos los únicos autores Diovis Robinet Morales, Yerandi Bracero Blanca y autorizamos a la facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de junio del 2009.

Yerandi Bracero Blanca.

_____
Firma del Autor

Diovis Robinet Morales.

_____
Firma del Autor

Ing. Darvis Dorvigny Dorvigny.

_____
Firma del Tutor

Ing. Iris Margarita Reina Heredia.

_____
Firma del Tutor

RESUMEN

En este trabajo se presenta el desarrollo de un sistema de pruebas de rendimiento realizadas a la centralita telefónica basada en software Asterisk, con el objetivo de medir su rendimiento en condiciones de sobrecarga de procesamiento. La información que procesa este sistema sirve a los operadores para la toma de decisiones en la implantación de un sistema telefónico.

Para el desarrollo del sistema se realizaron varios estudios sobre las diferentes aplicaciones que permiten realizar las pruebas a la SoftPBX Asterisk, o sistemas con características similares que existen hoy en el mundo. Se recogieron así los aspectos más importantes de estas herramientas que hicieron posible el desarrollo del sistema.

Algunos datos importantes que se manejan en las pruebas realizadas son la versión del sistema operativo, el modelo del CPU, los protocolos de señalización soportados por el sistema, los códecs usados en la prueba, además se mide el desempeño para una variedad de tipos de llamadas, entre otras. Durante estas llamadas se chequea la calidad percibida.

Para guiar el proceso de desarrollo del software se seleccionó la metodología XP y para la implementación del sistema se utilizó el lenguaje de programación Python, siguiendo el patrón arquitectónico Modelo-Vista-Controlador.

Palabras claves: Asterisk, Rendimiento, SoftPBX.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

A través de los años el hombre ha buscado diferentes formas de comunicarse, desde las señas hasta la comunicación a distancia por diferentes medios. La comunicación ha formado parte esencial en la vida de las personas, constituye la forma de transmitir sus pensamientos.

La comunicación ha evolucionado durante los últimos años. Cuando se pensaba que los únicos medios de difusión masiva serían la televisión y la radio, apareció Internet, la cual cambió todo tipo de concepción sobre las fuentes y las formas de comunicación. El uso de esta gran red de datos trajo consigo que se desarrollaran nuevas alternativas de gran impacto como la denominada tecnología VoIP (acrónimo de *Voice over IP*, Voz sobre IP).

VoIP constituye una nueva forma de trasmitir la voz utilizando una conexión a Internet en lugar de una línea telefónica común. Permite que se pueda hacer una llamada telefónica a través de la red de datos IP (acrónimo de *Internet Protocol*, Protocolo de Internet) y además enviar archivos, texto, identificador de llamadas personalizadas, entre otras funcionalidades.

Con la necesidad de compartir una o varias líneas telefónicas a un grupo de usuarios surgen las PBX (acrónimo de *Private Branch Exchange*, Central Telefónica Privada). Las PBX son capaces de redirigir las llamadas entrantes a uno o varios teléfonos. De la misma forma que un enrutador es responsable de dirigir los paquetes de un origen a su destino, una PBX es responsable de dirigir “llamadas telefónicas”

La evolución de las PBX y el desarrollo de la tecnología VoIP dieron lugar al desarrollo de las PBX basadas en software (SoftPBX), dando mayor usabilidad y flexibilidad a los sistemas de telecomunicaciones, convirtiéndose en la actualidad en uno de los negocios más competitivos.

El movimiento del software libre ha desarrollado una PBX basada en software de código abierto nombrada Asterisk. Es capaz de proporcionar todas las funcionalidades de una PBX tradicional de forma más flexible, e incluye nuevas características y funcionalidades. En los últimos años ha crecido notablemente el número de operadores que escogen la SoftPBX Asterisk como plataforma para brindar servicios de telefonía. Su rendimiento no está relacionado directamente con el número de usuarios que la utilizan, sino con la cantidad de llamadas simultáneas o concurrentes que tiene que procesar al realizar cada una de estas llamadas. Actualmente se desconoce la cantidad máxima de terminales que pueden establecer llamadas simultáneamente, y el comportamiento del servidor en condiciones de sobrecarga. Los operadores que escogen a la SoftPBX Asterisk necesitan conocer su comportamiento cuando se

realizan cada una de las llamadas telefónicas en condiciones de sobrecarga de procesamiento del servidor.

A partir del análisis de los problemas planteados, los esfuerzos estarán encaminados a resolver el siguiente **problema científico**: ¿Cómo medir el rendimiento de la SoftPBX Asterisk bajo condiciones de sobrecarga de procesamiento?

Teniendo en cuenta el problema científico planteado se define como **objeto de estudio**: SoftPBX Asterisk. Siendo el **campo de acción** del presente trabajo: Proceso de medición del rendimiento de la SoftPBX Asterisk.

Para el desarrollo de la investigación se traza como **objetivo general**: Desarrollar una herramienta capaz de medir el rendimiento de la SoftPBX Asterisk bajo condiciones de sobrecarga de procesamiento.

Para dar cumplimiento al objetivo anteriormente planteado se definen las siguientes **tareas investigativas**:

1. Instalación y configuración de la SoftPBX Asterisk, así como la búsqueda bibliográfica para estudiar su funcionamiento.
2. Realizar un estudio detallado de los protocolos de señalización que soporta Asterisk.
3. Realizar un estudio detallado de los códecs que soporta Asterisk.
4. Realizar un estudio de las distintas aplicaciones que son capaces de brindar información relevante sobre el rendimiento de la SoftPBX Asterisk.
5. Realizar un estudio de las principales técnicas y tecnologías actuales para el desarrollo de la aplicación.
6. Realizar un estudio sobre la metodología de desarrollo posible a utilizar.
7. Desarrollar una aplicación que brinde información relevante sobre el rendimiento de la SoftPBX Asterisk para la toma de decisiones en el dimensionamiento de un sistema telefónico.

La **idea a defender** del presente trabajo es: La utilización de la herramienta de prueba permitirá al operador obtener información relevante sobre el comportamiento de la SoftPBX Asterisk que sirva para la toma de decisiones en la implantación de un sistema telefónico en un determinado ambiente de trabajo.

Para dar cumplimiento a lo antes planteado se emplean los siguientes **métodos de investigación**:

Analítico–Sintético: Se utilizó durante el proceso de revisión bibliográfica para conocer el funcionamiento de Asterisk y de las diferentes aplicaciones que son capaces de brindar información relevante sobre el rendimiento de la misma. Al realizar el análisis de las principales técnicas, tecnologías, metodologías de

desarrollo de software para lograr la confesión de la aplicación. Además el análisis de los diferentes protocolos de señalización y códecs soportados por Asterisk, logrando así conocer los diferentes elementos que se relacionan con el objeto de estudio.
Observación-Experimento: Se registran los resultados las pruebas de rendimientos realizados a diferentes computadoras con Asterisk instalado, para su posterior estudio.

Para organizar el presente documento se ha dividido en 5 capítulos estructurados como sigue:
Capítulo 1: Fundamentación Teórica: En este capítulo se recogen los conceptos necesarios para comprender el desarrollo del sistema. Se realiza un estudio sobre las actuales aplicaciones que realizan la medición del rendimiento de la SoftPBX Asterisk. También se reflejan cuáles fueron las herramientas, los métodos y las tecnologías usadas en la solución del problema.
Capítulo 2: Características del Sistema: En este capítulo se aborda el objeto de automatización, la información que se maneja en el sistema a desarrollar, así como la arquitectura del sistema propuesto.
Capítulo 3: Exploración, Planificación de la Entrega e Iteraciones: En este capítulo se describen cada uno de los artefactos generados durante las fases de exploración y planificación del proyecto. Se define el alcance de cada iteración y se estima el tiempo necesario para desarrollar el producto. Finalmente se elabora un plan de entrega.
Capítulo 4: Implementación y Prueba: En este capítulo se definen las tareas de ingeniería por cada iteración y se realizan las pruebas de aceptación sobre el sistema.
Capítulo 5: Estudio de Factibilidad: En este capítulo se realiza el estudio de factibilidad para el sistema, se muestran los beneficios y se analizan los costos que representa la elaboración de la propuesta de solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción

En el presente capítulo se abordan las características y funcionalidades más importantes de la SoftPBX Asterisk, así como un análisis de las distintas aplicaciones que permiten la medición del rendimiento. Además se realiza un estudio de las técnicas, tecnologías y metodología de desarrollo de software a utilizar.

1.2 SoftPBX Asterisk.

Asterisk es una aplicación de software libre bajo licencia GPL (acrónimo de *General Public License*, Licencia Pública General) que proporciona funcionalidades de una PBX. Define las mismas características de una PBX tradicional pero es más flexible. Su nombre proviene del símbolo asterisco (*) en inglés.

Asterisk fue desarrollada con el objetivo de dar soporte a los usuarios en GNU/Linux. Mark Spencer es su creador y el principal desarrollador de las versiones más estables, actualmente promueve Asterisk junto a la empresa *Digium*, la cual se dedica al desarrollo de software y hardware de telefonía que trabajan para Asterisk. Originalmente fue creada para sistemas GNU/Linux pero actualmente funciona en una variedad de sistemas como OpenBSD, FreeBSD, Mac OS X, Solaris Sun y Windows. Pero GNU/Linux sigue siendo la distribución que más soporte presenta.

Asterisk incluye muchas características y funcionalidades, anteriormente sólo disponibles en caros sistemas propietarios PBX: buzón de voz, conferencias, IVR (acrónimo de *Interactive Voice Response*, Respuesta de Voz Interactiva), CTI (acrónimo de *Computer telephony integration*, Integración Teléfono Computadora), ACD (acrónimo de *Automatic Call Distributor*, Distribuidor Automático de Llamadas), y otras más. [1]

Los operadores pueden crear nuevas funcionalidades escribiendo un plan de discado (indicar las extensiones que se van a reservar en la centralita para cada uno de los servicios que se quiere incluir en la misma), en el lenguaje de script de Asterisk o añadiendo módulos escritos en cualquier otro lenguaje de programación soportado por GNU/Linux.

1.2.1 Protocolos de señalización soportado por Asterisk

Asterisk soporta diferentes tipos de protocolos de señalización tales como H.323, SIP, MGCP, SCCP y su protocolo nativo IAX actualmente se encuentra en la versión 2.0. A continuación se explican los protocolos antes mencionados: [2]

- **H.323** primer estándar VoIP desarrollado por la ITU (acrónimo de *International Telecommunication Union*, Unión Internacional de Telecomunicaciones) en 1996 con el objetivo de ofrecer un mecanismo de transporte para servicios multimedia sobre redes que no garantizan calidad de servicio, aunque su uso se ha extendido sobre todo al uso sobre redes IP. Fue el primer estándar en adoptar como medio de transporte el protocolo RTP (acrónimo de *Real-time Transport Protocol*, Protocolo de Transporte en Tiempo Real), siendo capaz de aplicar algoritmos de encriptación de la información, evitando de esta manera añadir elementos de seguridad adicionales a los requeridos para la conexión a Internet. Técnicamente es un protocolo potente y maduro, el interés por parte de los usuarios del protocolo y empresas actualmente ha disminuido debido principalmente a su complejidad. [2]
- **SIP** (acrónimo de *Session Initiation Protocol*, Protocolo de Inicio de Sesión) es un protocolo desarrollado por el IETF (acrónimo de *Internet Engineering Task Force*, Grupo de Tareas de Ingeniería de Internet) en 1999 para el control de llamadas multimedia, la implementación de servicios telefónicos avanzados, la modificación y terminación de sesiones de comunicación multimedia entre usuarios. Es el principal protocolo de señalización utilizado en el mundo de la VoIP. Estándar abierto y de grandes posibilidades. SIP está basado en HTTP (acrónimo de *HyperText Transfer Protocol*, Protocolo de Transferencia de Hipertexto) adoptando las características de este estándar como son la sencillez de su sintaxis y una estructura cliente/servidor basada en un modelo petición/respuesta. Otra de las ventajas de SIP es su sistema de direccionamiento, tiene una estructura parecida a la de un correo electrónico dotando a sus clientes de una alta movilidad facilitando una posible integración en comunicaciones móviles. El propósito de SIP es la comunicación entre dispositivos multimedia. SIP hace posible esta comunicación gracias a los protocolos RTP/RTCP que se usan para transportar los datos de voz en tiempo real, y SDP (acrónimo de *Session Description Protocol*, Protocolo Descripción de Sesión) se usa para la negociación de las capacidades de media. [2]
- **MGCP** (acrónimo de *Media Gateway Control Protocol*, Protocolo de Control de Pasarela y Multimedia) es otro estándar de señalización para VoIP desarrollado por la IETF. MGCP, está basado en un modelo cliente/servidor. De esta forma se consigue separar la señalización de la transmisión de la

información. Es un protocolo que está compuesto por tres componentes. Así la conversión del contenido multimedia es realizada por el MG (*Media Gateway*), el control de la señalización del lado IP es realizada por el MGC (*Media Gateway Controller*), y el control de la señalización del lado de la red de Conmutación de Circuitos es realizada por el SG (*Signalling Gateway*). Es óptimo para los grandes operadores de telefonía. MGCP es igual al protocolo SIP que usa SDP para describir y negociar capacidades de media. Protocolo de VoIP de arquitectura muy compleja. [2]

- **SCCP** (acrónimo de *Skinny Client Control Protocol*). Protocolo propietario de Cisco para la gestión entre los teléfonos y su servidor de VoIP. Utiliza TCP/IP (acrónimo de *Transmission Control Protocol*, *Internet Protocol*, Protocolo de Transmisión de Información, Protocolo de Internet) para conectarse a los *Call Managers* en un cluster. Para el tráfico de datos utiliza RTP/UDP/IP. Los clientes son teléfonos IP, que no necesitan de mucha memoria ni procesamiento. El servidor es el que aprende las capacidades de los clientes, controla el establecimiento de la llamada, envía señales de notificación, y reacciona ante señales del cliente. El servidor usa SCCP para comunicarse con los clientes y si la llamada sale por un gateway, usa H323, MGCP o SIP. [2]
- **IAX2** (acrónimo de *Inter Asterisk Exchange*, Protocolo de Intercambio entre servidores Asterisk) es la segunda versión del protocolo IAX. El protocolo original IAX ha quedado obsoleto en favor de IAX2. Fue desarrollado por la empresa Digium, inicialmente diseñado para la comunicación entre sistemas Asterisk remotos. Actualmente es empleado también entre servidores y clientes VoIP. Desde su creación se ha revelado como un protocolo robusto, potente y flexible. Numerosos fabricantes de hardware lo implementan en sus equipos, aunque no es un protocolo estandarizado hasta el momento. El principal objetivo de IAX2 es minimizar el ancho de banda. La estructura básica de IAX2 se fundamenta en la multiplexación de la señalización y del flujo de datos sobre un simple puerto UDP, normalmente por el 4569, tanto la información de señalización como los datos viajan conjuntamente y por tanto lo hace menos proclive a problemas de NAT (acrónimo de *Network Address Translation*, Traducción de Dirección de Red), permitiéndole pasar los routers y firewalls de manera más sencilla. Permite manejar una gran cantidad de códecs y transportar cualquier tipo de datos. Todas las características de IAX2 se deben a que su diseño se basó en estándares de señalización y de transmisión de datos, quedándose solo con lo mejor de cada uno. Algunos protocolos tomados como base fueron: SIP, MGCP y RTP. [2]

1.2.2 Códecs de voz soportado por Asterisk

La elección de un códec a utilizar por parte de los usuarios u operadores puede basarse principalmente en el ancho de banda del que se disponga. A la hora de establecer una conexión, se suele negociar qué tipo de códec se va a utilizar. A continuación se describen características de algunos códecs que más se utilizan en Asterisk: [3]

Códec	**Ancho de Banda Requerido**	**Características**
G.711	64 kbps.	Versión libre y se encuentra estandarizado por la ITU. Conocido como α-law (versión de Europea) μ –law (versión de EE.UU). Diseñado para entregar la máxima calidad de voz sin compresión; muy bajo consumo del CPU.
G.723.1	5.3/6.3 kbps.	Comúnmente usado por proveedores de VoIP. Requiere de licencia para su uso. Compresión muy alta manteniendo excelente calidad de audio. Uso intenso del CPU.
G.726	16/24/32/40 kbps.	Versión libre, mejorada de G.721 y G.723). Se utiliza en troncales internacionales para ahorrar ancho de banda. Buen nivel de compresión con poco uso de CPU.
G.729a	8 kbps.	Se encuentra estandarizado por la ITU. Excelente relación ancho de banda calidad. Requiere de licencia para su uso. Es un estándar de bajo consumo de ancho de banda.
GSM	13 kbps.	Versión libre y es usado en la redes celulares GSM. Alto porcentaje de compresión; se usa en hardware y software disponibles en el mercado de VoIP. El Uso del CPU es intenso.
LPC-10	2.5 kbps.	Mínimo ancho de banda. Utilizado en la voz robótica. El uso del CPU es intenso.

ILBC	13.33 / 15.2 kbps.	Es libre y apropiado para las comunicaciones de voz robustas sobre IP. Robusto ante pérdida de paquetes. Requiere de mucho procesamiento que impide tener varias llamadas a la vez.
SPEEX	2.15 a 44.2 kbps.	Gran flexibilidad soporta codificaciones estéreo, manejo de paquetes perdidos y detención de actividad de voz. Es gratuito para la compresión de audio. El uso de CPU es intenso.

Tabla 1: Códecs soportado por Asterisk.

Se debe elegir el códec que más se adapte a las necesidades de los usuarios. Si se tiene un buen ancho de banda disponible se puede evitar la compresión.

1.2.3 Consideraciones sobre el rendimiento de Asterisk.

Los operadores a la hora de escoger a la SoftPBX Asterisk como plataforma telefónica deben tener presentes diferentes consideraciones que influyen en su rendimiento y correcto funcionamiento. Algunas de esas consideraciones son comentadas de manera breve en los puntos siguientes [4] [7]:

- Asterisk tiene una limitante importante: depende mucho del hardware donde será instalado el servidor; esto realmente constituye un punto crítico. Es importante elegir buenos componentes de hardware que ofrezcan compatibilidad y estabilidad con Asterisk. El operador que usará Asterisk debe conocer los componentes a nivel de arquitectura, chipsets, rendimiento, estabilidad, disponibilidad y compatibilidad.
- Para Asterisk es recomendable usar como sistema base GNU/Linux porque sigue siendo la el sistema operativo que más soporte presenta para esta tecnología. Del sistema operativo se debe entender no solo su instalación y configuración, sino el afinamiento para garantizar la máxima estabilidad y el mejor manejo de sus recursos. Una simple instalación de valores por defecto no necesariamente se ajustará a las necesidades de los usuarios. Se debe tener presente cuáles características deberá proporcionar el sistema para su funcionamiento.
- Es recomendable preservar al máximo el servidor donde se encuentra instalada la SoftPBX Asterisk, hasta el punto que no debe estar corriendo en la PC otras aplicaciones que puedan comprometer el rendimiento del servidor.

- Es recomendable tener experiencia en los diferentes protocolos de señalización que el usuario necesitará para su comunicación H.323, SIP, IAX2, entre otros, así como en los diferentes códecs que se usarán para codificar y decodificar los datos de voz de las llamadas que pasan por el servidor.
- Se debe tener presente la versión de Asterisk que el operador desea instalar, así como conocer en detalles los componentes y submódulos que se quieren poner en marcha en su funcionamiento.

Tener en cuenta todos los aspectos antes mencionados ayudará a darle una mayor validez de selección una vez que el operador se decida a utilizar Asterisk como una planta telefónica de software. En realidad el tamaño de un sistema Asterisk no viene dado por el número de los usuarios o grupos que maneja, sino más bien por el número de llamadas simultáneas que puede soportar. Si se relacionan bien y se combinan cada uno de los aspectos anteriores a las consideraciones de los operadores, se podrán tener buenos resultados.

1.3 Aplicaciones actuales para la medición del rendimiento de Asterisk

En el mundo hoy existen algunas aplicaciones que permiten medir el rendimiento de la SoftPBX Asterisk. A continuación se explican brevemente algunos de estos software:

- **SIPp** es una aplicación de software libre que genera tráfico de llamadas para realizar pruebas, pero solo para el protocolo SIP, lo cual constituye una dificultad cuando existen otros protocolos de señalización que soporta Asterisk. Incluye varios escenarios que establecen y liberan múltiples llamadas. Durante su ejecución muestra estadísticas acerca de la prueba que está siendo ejecutada como el total de llamadas realizadas a ejecutar durante un período determinado, el servidor remoto que se le realizan las pruebas y la cantidad de llamadas concurrentes soportadas. También es capaz de enviar tráfico de audio y vídeo a través de RTP. Otra de las desventajas que presenta es que solo garantiza la generación de llamadas. [5]
- **Astertest** es una aplicación desarrollada en plataforma Windows. Sirve para poner a prueba la carga de la CPU de la SoftPBX Asterisk que se desea probar. Para realizar las pruebas se debe tener un servidor Asterisk que será el que originará las llamadas, más el servidor Asterisk que se le desea realizar las pruebas. Es una buena aplicación pero tiene la dificultad que funciona solo sobre plataforma propietaria.[6]
- **PBX-Test** es una aplicación desarrollada en lenguaje de programación PHP (acrónimo de *Hypertext Preprocessor)*). Para realizar las pruebas se necesita de otra máquina con Asterisk, más

tener instalado PHP en él para generar llamadas contra el servidor que se le quieren realizar las diferentes pruebas. Permite escribir un conjunto de instrucciones en PHP para emular las acciones del cliente al azar. Solo es usada para generar llamadas. [7]

En la actualidad el país no cuenta con aplicaciones que permitan realizar las pruebas a la SoftPBX Asterisk. Se recogieron así los aspectos más importantes de estas herramientas que hicieron posible el desarrollo del sistema.

1.4 Lenguaje Programación.

Los lenguajes de programación son herramientas que nos permiten crear programas y software. Están constituidos por un grupo de reglas gramaticales y símbolos utilizables. A continuación se describe el lenguaje de programación seleccionado para la confección de la herramienta de prueba y se fundamenta el porqué de su elección. .

1.4.1 Python.

Lenguaje de programación creado por Guido Van Rossum finales de los años 90, motivado por la experiencia en la creación de otro lenguaje de programación llamado ABC. El objetivo de Rossum era cubrir la necesidad de un lenguaje orientado a objetos de uso sencillo que sirviese para tratar diversas tareas dentro de la programación que habitualmente se hacía en Unix usando lenguaje de programación C. Entre sus ventajas de pueden destacar: [8]

- Es multiplataforma por lo que el mismo código funciona en cualquier arquitectura, la única condición es que disponga del intérprete del lenguaje. No es necesario compilar el código una vez para cada arquitectura.
- Permite dividir el programa en módulos reutilizables desde otros programas Python.
- Contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas del propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.
- Incluye módulos que proporcionan entrada y salida de ficheros, llamadas al sistema, sockets y hasta interfaces gráficas como Tk, GTK, Qt, WxPython entre otras.
- La sencillez y velocidad con la que se crean los programas. Un programa en Python puede tener de 3 a 5 líneas de código menos que su equivalente en Java o C.

- Posee una licencia de código abierto, denominada FSF (acrónimo de *Python Software Foundation License)*, que es compatible con la licencia GPL.

1.5 Herramientas Utilizadas.

En la actualidad existen numerosas alternativas para disminuir la carga laboral de los programadores, sobre todo en la generación automática de código que permite este propósito. A continuación se describen las herramientas a utilizar para la confección del sistema de prueba.

1.5.1 Eclipse.

Eclipse es un IDE (acrónimo de *Integrated Development Environment,* Ambiente de Desarrollo Integrado*)* escrito en lenguaje de programación Java, que fue liberado por la IBM (IBM: empresa que fabrica y comercializa hardware, software y servicios relacionados con la informática) a la comunidad del software libre su objetivo inicial era el java, la comunidad le ha ido incluyendo algunos plugins (plugins: plug-in -en inglés "enchufar"-, también conocido como addin, add-in, addon o add-on) para otros lenguajes de programación. En el caso particular para desarrollar en lenguaje de programación Python se creó Pydev. PyDev es un plugins para el Eclipse el cual funciona perfectamente bajo GCJ (acrónimo de *GNU Compiler for Java*, Compilador de java de GNU).

PyDev tiene algunas características que se relacionan a continuación: [10]

- Resaltado de sintaxis, explorador de clases y resaltado de errores, lo básico de todo IDE. El resaltado de errores se hace utilizando PyLint. Se pueden detectar errores que van desde un objeto no declarado hasta prácticas feas de programación como líneas muy largas o nombres de variable muy cortos.
- Completar código no es más que el clásico asistente que ayuda sugiriendo qué se puede colocar digamos después de un punto. Muy útil a la hora de recordar nombres de métodos.
- Depurador gráfico bastante fácil de usar permite al usuario depurar una buena perspectiva gráfica de todo lo que está sucediendo. Con Eclipse y PyDev hacen un excelente trabajo en ese sentido.

1.5.2 WxGlade.

WxGlade es un editor de interfaz gráfica escrita en Python con el popular GUI toolkit WxPython, que le ayuda a crear wxWidgets (bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas muy potables entre distintos sistemas operativos). Este programa permite el diseño práctico de las ventanas y

generar código fuente en varios lenguajes, incluyendo Python, C+ +, Perl entre otros. Solo es usado para diseñar el código generado; no hace nada aparte de mostrar los wxWidgets creados. Permite al usuario contar con una herramienta fácil en la creación de entornos visuales en Python [11].

1.6 Patrón de Arquitectura (MVC).

La arquitectura MVC (acrónimo de *Model View Controller,* Modelo, Vista y Controlador) fue introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk [12].

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. En la figura 1 se muestra la arquitectura MVC

- Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- Vista: Esta presenta el modelo en un formato adecuado para interactuar, usualmente es la interfaz de usuario.
- Controlador: Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

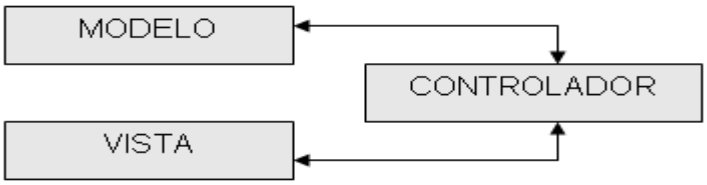


Figura 1: Patrón MVC.

Este modelo de arquitectura presenta varias ventajas:

• Hay una clara separación entre los componentes de un programa; lo cual permite implementarlos por separado y luego unirlas en tiempo de ejecución

• Hay un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.

• La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución.

1.7 Metodología Desarrollo de Software.

En un proyecto de desarrollo de software la metodología define “Quién” debe hacer, “Qué”, “Cuándo” y “Cómo” hacer [13]. La metodología constituye la columna vertebral del proceso de desarrollo de software. No existe una metodología universal. Las características de cada proyecto (equipo de desarrollo, recursos, entre otros) exigen que el proceso sea configurable.

Las metodologías les permiten a los ingenieros guiar con mayor precisión su trabajo para lograr productos de software bien acabados y que cumplan las métricas de calidad establecidas. Para el desarrollo de cualquier producto de software es importante la búsqueda de la metodología que más se adapte a las condiciones con que se cuenta para desarrollar determinados proyectos (requerimientos, límite de tiempo, calidad, entre otros). La elección respecto a la utilización o no de una determinada metodología depende principalmente del grado de predictibilidad que se desee tener en el desarrollo.

Existen numerosas propuestas metodológicas entre las que se encuentran las metodologías pesadas o tradicionales y las ágiles o ligeras.

Las metodologías tradicionales ponen gran énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos que se deben producir, incluyendo modelado y documentación detallada [14]. Este enfoque tradicional es conveniente para proyectos que requieran mucho tiempo y recurso, pero no resulta muy adecuado para proyectos donde el entorno del sistema es muy cambiante y es necesario reducir los tiempos de desarrollo.

Las metodologías ágiles están orientadas especialmente a proyectos pequeños y permiten que los programadores se concentren solamente en aquellas funciones que se necesitan inmediatamente. Se realizan entregas al cliente frecuentemente, de las cuales se obtiene retroalimentación constante, posibilitando respuestas rápidas a los cambios en el negocio [14].

La siguiente tabla muestra las diferencias entre las metodologías ágiles y tradicionales [14]:

Metodologías Ágiles	**Metodologías Tradicionales**
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguido por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.

Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 2: Comparación entre las metodologías ágiles y las tradicionales.

De acuerdo a las características que promueven los objetivos especificados en este trabajo, se optó por elegir una metodología ágil para el desarrollo del sistema. A continuación se explican cada una de las características del porqué de su elección:

Debido a que se dispone de poco tiempo para el desarrollo del software y poca disponibilidad del personal se hace imposible la existencia de muchos roles para el desarrollo del sistema. El sistema a desarrollar es realizado por dos programadores, posibilitando así la menor tasa de errores, mejor diseño y mayor satisfacción, permitiendo la posibilidad de cambiar cualquier parte del código en cualquier momento. Existe una buena comunicación entre los clientes y los programadores del sistema, la colaboración con el cliente es más que un contrato. El cliente es parte del equipo de desarrollo. Es una necesidad producir versiones del sistema que sean operativas en el menor tiempo posible, más que conseguir una buena documentación, solo documentos que sean necesarios para tomar decisiones inmediatas. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software desde un par de semanas o un par de meses con el menor tiempo posible de entregas. Lo importante es que el software funcione en la medida del progreso.

1.7.1 Programación Extrema (Extreme Programming, XP).

Cuando se le quiere dar mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas se hace necesario conocer sobre metodologías ágiles.

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios [15].

XP define los siguientes roles [14]:

- *Programador*. El programador escribe las pruebas unitarias y produce el código del sistema.
- *Cliente*. Escribe las Historias de Usuario y las pruebas funcionales para validar su implementación. Asigna la prioridad a las Historias de Usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- *Encargado de pruebas (Tester).* Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- *Encargado del seguimiento (Tracker*). Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- *Entrenador (Coach).* Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- *Consultor*. Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- *Gestor (Big boss)*. Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje.

El Proceso XP es:

1) El cliente define el valor de negocio a implementar.
2) El programador estima el esfuerzo necesario para su implementación.
3) El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4) El programador construye ese valor de negocio.

5) Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

Las Prácticas XP:

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas.

- El juego de la planificación.
- Entregas pequeñas.
- Diseño simple.
- Pruebas.
- Programación en parejas.
- Propiedad colectiva.
- Cliente in-situ.
- Estándares de programación.

Teniendo en cuenta las características antes expuestas y ciclo de vida de desarrollo que propone la metodología XP se decide utilizar esta como metodología de desarrollo de software.

1.8 Conclusiones

En este capítulo se describieron algunos conceptos relacionados con el dominio del problema. También se realizó un estudio sobre las aplicaciones que permiten realizar las diferentes pruebas a la SoftPBX Asterisk, o sistemas con características similares. Se describió además el lenguaje de programación que se escogió para la realización del sistema, así como las herramientas utilizadas donde se pudieron

apreciar las principales características de las mismas y por último se hizo un estudio de las metodologías de desarrollo de software en el cual se seleccionó la metodología XP para guiar el proceso de desarrollo.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

Después de abordar en el capítulo anterior los diferentes conceptos para la comprensión del dominio del problema, las técnicas y las herramientas que permiten realizar las pruebas a la SoftPBX Asterisk, se está en condiciones de comenzar a desarrollar el sistema. En el siguiente capítulo se especifica el objeto de automatización, la información que se maneja y la propuesta del sistema.

2.2 Objeto de estudio

2.2.1 Objeto de automatización

Proceso de medición del rendimiento de la SoftPBX Asterisk.

2.2.2 Información que se maneja

- Versión del Sistema Operativo.
- Versión del kernel del Sistema Operativo.
- Modelo del CPU.
- Porciento uso del CPU.
- Porciento uso de la Memoria RAM.
- Memoria RAM total del sistema.
- Memoria RAM usada del sistema.
- Memoria RAM libre del sistema.
- Swap total del sistema.
- Swap usada del sistema.
- Swap libre del sistema.
- Espacio total del disco duro.
- Espacio usado del disco duro.
- Espacio libre del disco duro.
- Tipo de prueba.
- Protocolos señalización usado en la prueba.
- Códec usado en la prueba.
- Cantidad de llamadas que se están ejecutando en la prueba.

- Cantidad de canales activos que se están ejecutando en la prueba.

2.3 Propuesta de sistema

Para darle solución al problema planteado se propone una aplicación de escritorio con el objetivo de crear un ambiente amigable al operador que usara la herramienta de prueba. La aplicación se realiza utilizando lenguaje de programación Python por las características que este posee. A continuación se expondrá la arquitectura del sistema.

2.3.1 Arquitectura del sistema propuesto

Para el desarrollo de software es necesario establecer una estructura de los componentes que lo integran de manera que soporte las funcionalidades requeridas.

La arquitectura candidata para el desarrollo del sistema se puede observar en la **Figura 2**, donde se muestran sus componentes principales, la forma en que éstos deben interactuar y coordinar para llevar a cabo el objetivo del sistema.

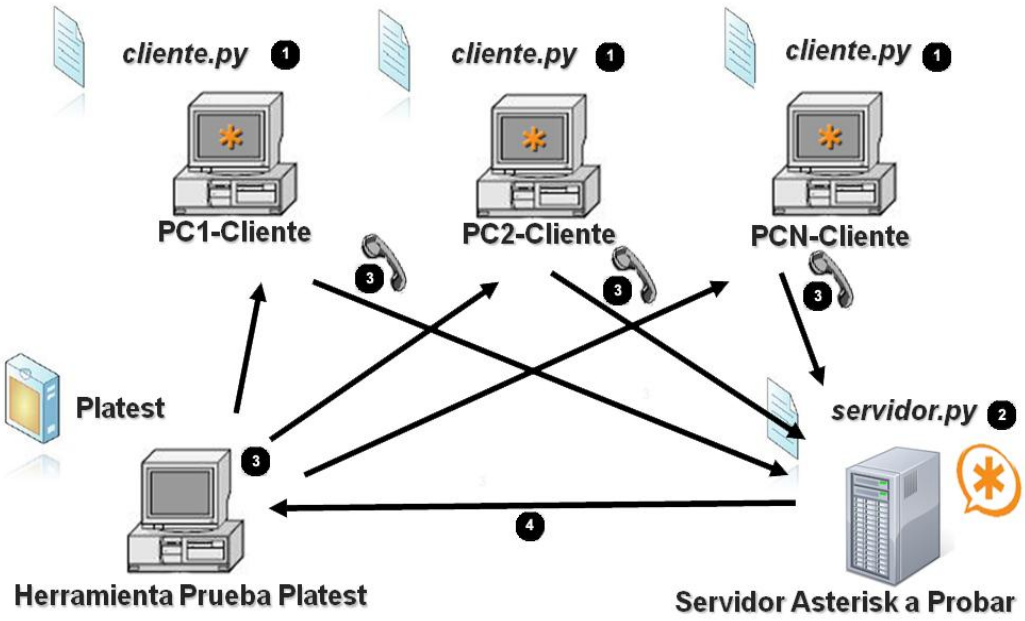


Figura 2: Arquitectura candidata.

Para realizar las pruebas se necesita de una o varias PC clientes según las seleccionadas por el operador que tengan instaladas la SoftPBX Asterisk, las cuales permitirán generar llamadas contra el servidor a

probar. También se necesita tener instalada la herramienta de prueba Platest (nombre de la aplicación a desarrollar) en una PC que se escogió como cliente o en otra PC disponible; y dónde se encuentra instalado el servidor Asterisk que se desea probar.

Después de garantizados todos los aspectos anteriores, se está en condición de comenzar a realizar las pruebas:

1) En cada una de las PC clientes se instalará un script *cliente.py* desde la aplicación Platest especificándole cada uno de los aspectos necesarios para realizar las llamadas contra el servidor a probar.
2) Después de haber configurado los scripts *cliente.py* mediante la aplicación Platest se pasa a la instalación y configuración del script *servidor.py* de la PC donde se encuentra el servidor Asterisk a probar, según la configuración de cada uno de los scripts *cliente.py*.
3) Una vez instalados y configurados los scripts *cliente.py* y *servidor.py* desde la aplicación Platest se le da la orden a los servidores clientes que comiencen a generar las llamadas contra el servidor a probar.
4) El servidor a probar mediante el script *servidor.py* comenzará a ofrecerle información relevante a la aplicación Platest inmediatamente después de haber comenzado el proceso de generación de llamadas.

2.4 Conclusiones.

En este capítulo se realizó una descripción de la solución propuesta para la realización del sistema y se detalló la arquitectura del sistema propuesto. A partir de este momento se está en condiciones de comenzar a construir el sistema que constituye la propuesta de solución.

CAPÍTULO 3: EXPLORACIÓN, PLANIFICACIÓN DE LA ENTREGA DE ITERACIONES.

3.1 Introducción.

En el presente capítulo se aborda el trabajo realizado en las fases de exploración y planificación pertenecientes a la metodología de desarrollo XP utilizada para el desarrollo del sistema propuesto. Además se exponen los artefactos generados durante el transcurso de dichas fases.

3.2 Fase de Exploración.

La Fase de exploración es la primera fase de desarrollo de la metodología XP. Durante la misma se realiza el proceso de identificación de las Historias de Usuarios, así como la familiarización de los equipos de trabajo con las tecnologías y herramientas seleccionadas para la construcción del proyecto para lograr la primera entrega del producto.

3.2.1 Historias de Usuarios.

Las Historias de Usuarios son las técnicas utilizadas en XP para especificar los requisitos del software. Estas se escriben desde la perspectiva del cliente donde describen brevemente las características que el sistema debe de realizar aunque también los desarrolladores pueden brindar también su ayuda en la identificación de las mismas. El contenido de estas debe ser concreto, sencillo, dinámico y flexible. Cada Historia de Usuario es lo suficientemente comprensible, escritas en un lenguaje natural para que los programadores puedan implementarla en unas semanas.

Durante la fase de exploración se identificaron cinco Historias de Usuarios, las cuales se detallan a continuación:

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario:** Configurar Script Cliente.
Riesgo en Desarrollo: Alto.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta.	**Puntos Estimados:** 1
Programador Responsable: Yerandi Bracero Blanca.	
Descripción: Se configuran los *script* de las PC clientes con los datos enviados desde la	

aplicación Platest. Los datos los introduce el operador en la aplicación. La configuración de los script permite que se generen llamadas de prueba hacia el servidor de la SoftPBX Asterisk. En el script se configuran los siguientes datos: • Tipo de prueba. • Protocolo señalización usado en la prueba. • Códec usado en la prueba. • Cantidad de llamadas a realizar en la prueba. • Servidor a probar.
Observaciones:

Tabla 3: Historia de Usuario Configurar Script Cliente.

Historia de Usuario	
Número: 2	**Nombre Historia de Usuario:** Configurar Script Servidor.
Riesgo en Desarrollo: Alto.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta.	**Puntos Estimados:** 1
Programador Responsable: Diovis Robinet Morales.	
Descripción: Se configura el script en el servidor que se quiere probar según la configuración de los script de los clientes configurados para permitir la realización de las llamadas de los clientes contra el servidor de la SoftPBX Asterisk.	
Observaciones:	

Tabla 4: Historia de Usuario Configurar Script Servidor.

Historia de Usuario	
Número: 3	**Nombre Historia de Usuario:** Generar llamadas.

Riesgo en Desarrollo: Alto.	**Iteración Asignada:** 2
Prioridad en Negocio: Alta.	**Puntos Estimados:** 2
Programador Responsable: Yerandi Bracero Blanca	
Descripción: Los clientes configurados comienzan a generar las llamadas hacia el servidor al cual se le quiere realizar las pruebas.	
Observaciones:	

Tabla 5: Historia de Usuario Generar llamadas.

Historia de Usuario		
Número: 4	**Nombre Historia de Usuario:** Obtener información de Prueba.	
Riesgo en Desarrollo: Alto.		**Iteración Asignada:** 2
Prioridad en Negocio: Alta.		**Puntos Estimados:**2
Programador Responsable: Diovis Robinet Morales.		
Descripción: Una vez que se generan las llamadas de prueba se comienza a obtener información cada un intervalo de tiempo de 1 segundo sobre lo que realmente está ocurriendo en el servidor que se está probando. En cada intervalo de tiempo se obtiene y muestra la siguiente información: • Versión del sistema operativo. • Versión del kermel del sistema operativo. • Modelo de la CPU. • Porciento uso del CPU. • Porciento uso de la Memoria RAM. • Memoria RAM total del sistema. • Memoria RAM usada del sistema. • Memoria RAM libre del sistema.		

• Swap total del sistema. • Swap usada del sistema. • Swap libre del sistema. • Espacio total del disco duro. • Espacio usado del disco duro. • Espacio libre del disco duro. • Cantidad de llamadas que se están ejecutando en la prueba. • Cantidad de canales activos que se están ejecutando en la prueba. De la información obtenida se visualiza en una gráfica el porciento de uso del CPU y la memoria RAM del sistema.
Observaciones:

Tabla 6: Historia de Usuario Obtener Información de Prueba.

Historia de Usuario		
Número: 5	**Nombre Historia de Usuario:** Mostrar Reporte General.	
Riesgo en Desarrollo: Alto.		**Iteración Asignada:** 3
Prioridad en Negocio: Alta.		**Puntos Estimados:** 2
Programador Responsable: Diovis Robinet Morales, Yerandi Bracero Blanca.		
Descripción: Se muestra un reporte general al operador de la información de prueba obtenida cuando se realizaban las diferentes llamadas al servidor.		
Observaciones:		

Tabla 7: Historia de Usuario Mostrar Reporte General.

3.3 Planificación.

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada Historia de Usuario. La medida para calcular el esfuerzo necesario es el punto. Un punto se considera

como una semana ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción, generalmente una Historia de Usuario no excede los 3 puntos.

3.3.1 Estimación de esfuerzo por Historia de Usuario.

Para el desarrollo de la aplicación propuesta en este trabajo se realizó una estimación del esfuerzo necesario para realizar cada una de las Historias de Usuarios identificadas, los cuales se muestran en la siguiente tabla:

Historia de Usuario	Puntos Estimados
Configurar Script Cliente.	1
Configurar Script Servidor.	1
Generar llamadas.	2
Obtener información de Prueba.	2
Mostrar Reporte General.	2

Tabla 8: Estimación de esfuerzos por Historia de Usuario.

3.3.2 Iteraciones.

Una vez identificadas las Historias de Usuarios del sistema y estimado el esfuerzo dedicado a la realización de cada una de estas se procede a la planificación de la etapa de implementación de la aplicación. Para la realización de la implementación de la aplicación del sistema se dividió en tres iteraciones, las cuales se describen a continuación:

Primera Iteración: En esta iteración se implementa la Historia de Usuario número uno y número dos de alta prioridad, al concluir se contará con una primera versión (0.1) que será mostrada al cliente, como versión inicial de prueba, para comprobar si satisface sus expectativas. La realización de las mismas va dando una idea de cómo quedará la aplicación aunque todavía estará en sus inicios.

Segunda Iteración: En esta iteración se implementa la Historia de Usuario número tres y número cuatro de alta prioridad , al concluir dicha iteración se contará con una nueva versión del producto (0.2), a la cual se le realizarán pruebas para verificar si cumple con las funcionalidades requeridas por el cliente. El cliente observará dichas pruebas para comprobar si realmente cumple con sus expectativas. La realización de las mismas va dando una idea más completa de la aplicación en cuestión.

Tercera Iteración: En esta iteración se implementa la Historia de Usuario número cinco de alta prioridad, al finalizarla se contará con la primera versión del producto final (1.0) y se realizarán pruebas al sistema completo para definir si cumple con todos los requerimientos necesarios del cliente. La realización de la misma da el fin a la implementación de la aplicación.

3.3.3 Plan de duración de las Iteraciones.

Como parte del ciclo de vida del proyecto usando metodología XP se crea el plan de duración de las iteraciones, mostrando: las iteraciones involucradas en el proceso de desarrollo, las Historias de Usuarios a desarrollar en cada iteración y el tiempo de duración estimado para cada una. La siguiente tabla muestra el orden que poseen las Historias de Usuarios, que será el orden a seguir para la implementación de las mismas.

Iteraciones	Orden de las Historias Usuario a implementar	Duración de la estimación
Iteración 1	Configurar Script Cliente.	1 semana.
Iteración 1	Configurar Script Servidor.	1 semana.
Iteración 2	Generar llamadas.	2 semanas.
Iteración 2	Obtener información de Prueba.	2 semanas.
Iteración 3	Mostrar Reporte General.	2 semanas.

Tabla 9: Plan de duración de las iteraciones.

3.3.4 Plan de entrega.

A continuación se presenta el plan de entregas ideado para la fase de implementación. Al final de cada iteración se le realizarán pruebas al producto obtenido, en la fecha indicada en la siguiente tabla:

Herramienta	Final 1ra iteración 1ra semana de marzo	Final 2da iteración 3ra semana de marzo	Final 3ra iteración 1ra semana de abril
Platest	0.1	0.2	1.0

Tabla 10: Plan de entregas.

3.5 Conclusiones

En el presente capítulo se abordó todo lo referente a las fases de exploración, planificación de la entrega e iteraciones del proyecto, donde se describieron cada uno de los artefactos generados en las mismas

durante su desarrollo, dando una idea de la magnitud del proyecto a realizar. Al culminar la última iteración descrita en el capítulo, el sistema estará listo para entrar en producción.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA.

4.1 Introducción.

En el presente capítulo se detallan las tres iteraciones llevadas a cabo durante la etapa de construcción del sistema, exponiendo las tareas generadas por cada Historia de Usuario, así como las pruebas de aceptación efectuadas sobre el proyecto. Es válido aclarar que para el diseño y construcción de las aplicaciones, la metodología XP no requiere la representación del sistema mediante diagramas de clases y componentes utilizando notación UML aunque puede implementarse su uso siempre que sea necesario para mantener una buena comunicación entre el cliente y los desarrolladores. Con este fin se crearon los diagramas antes mencionados, estos pueden verse en los anexos 1 y 2.

4.2 Primera Iteración.

Durante esta iteración se abordaron las Historias de Usuarios de mayor prioridad donde se construyó la base de la arquitectura del producto con las funcionalidades primarias necesarias para ser mostrado al cliente y obtener una rápida y amplia retroalimentación. A continuación se describen en la siguiente tabla:

Historia de Usuario	Estimación	Real
Configurar Script Cliente.	1	1
Configurar Script Servidor.	1	1

Tabla 11: Historias de Usuarios implementadas en la primera iteración.

4.2.1 Tareas de Historia de Usuario Abordadas en la iteración.

Las siguientes tablas han sido organizadas, los datos referentes a las Tareas de Ingeniería de la primera iteración.

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Creación de la interfaz para Configurar Script Cliente.	

Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 0.5
Fecha Inicio: 1 de marzo 2009.	**Fecha Fin:** 4 de marzo 2009.
Programador Responsable: Yerandi Bracero Blanca.	
Descripción: Se crea la interfaz para configurar script cliente.	

Tabla 12: Tarea de Ingeniería Nro. 1 para la Historia de Usuario Configurar Script Cliente.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Inserción de los datos referentes para la creación del Script Cliente.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 0.5
Fecha Inicio: 4 de marzo 2009.	**Fecha Fin:** 7 de marzo 2009.
Programador Responsable: Yerandi Bracero Blanca.	
Descripción: Se le envían los datos desde la aplicación Platest.	

Tabla 13: Tarea de Ingeniería Nro. 2 para la Historia de Usuario Configurar Script Cliente

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 2
Nombre Tarea: Creación de la interfaz para Configurar Script Servidor.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 0.5
Fecha Inicio: 1 de marzo 2009.	**Fecha Fin:** 5 de marzo 2009**.**
Programador Responsable: Diovis Robinet Morales.	

Descripción: Se crea la interfaz para configurar script servidor.

Tabla 14: Tarea de Ingeniería Nro. 1 para la Historia de Usuario Configurar Script Servidor

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre Tarea: Inserción de los datos referentes para la creación del Script Servidor.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 0.5
Fecha Inicio: 5 de marzo 2009.	**Fecha Fin:** 8 de marzo 2009.
Programador Responsable: Diovis Robinet Morales.	
Descripción: Se le envían los datos desde la aplicación Platest según la configuración de los script clientes.	

Tabla 15: Tarea de Ingeniería Nro. 2 para la Historia de Usuario Configurar Script Servidor

4.3 Segunda Iteración.

Durante esta iteración se abordaron las Historias de Usuarios de mayor prioridad donde se definen las tareas de desarrollo de las mismas. Al culminar se consta de un producto casi listo para su puesta en funcionamiento con la mayoría de sus funcionalidades más críticas ya implementadas. A continuación se describe en la siguiente tabla:

Historia de Usuario	Estimación	Real
Generar llamadas.	2	2
Obtener información de Prueba.	2	2

Tabla 16: Historias de Usuarios abordada en la segunda iteración.

4.3.1 Tareas de Historia de Usuario Abordadas en la iteración.

En las siguientes tablas han sido organizados los datos referentes a las Tareas de Ingeniería de la segunda iteración.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 3
Nombre Tarea: Inserción de los datos referentes para generar las llamadas.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 7 de marzo 2009.	**Fecha Fin:** 21 de marzo 2009.
Programador Responsable: Yerandi Bracero Blanca.	
Descripción: Se les envían los datos desde la aplicación Platest a los clientes para permitir la realización de las llamadas de los clientes contra el servidor de la SoftPBX Asterisk.	

Tabla 17: Tarea de Ingeniería Nro. 1 para la Historia de Usuario Generar Llamadas.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 4
Nombre Tarea: Creación de la interfaz de la información de Prueba.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 8 de marzo 2009.	**Fecha Fin:** 15 de marzo 2009.
Programador Responsable: Diovis Robinet Morales.	
Descripción: Se crea la interfaz de información de prueba.	

Tabla 18: Tarea de Ingeniería Nro. 1 para la Historia de Usuario Obtener información de Prueba.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 4
Nombre Tarea: Obtener los datos referente a la prueba.	

Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 15 de marzo 2009.	**Fecha Fin:** 22 de marzo 2009.
Programador Responsable: Diovis Robinet Morales.	
Descripción: Se obtienen los datos de la prueba y se procesan mostrándoselos al Usuario en la interfaz.	

Tabla 19: Tarea de Ingeniería Nro. 2 para la Historia de Usuario Obtener información de Prueba.

4.4 Tercera Iteración.

En el transcurso de esta iteración se implementan las Historias de Usuarios referente a la tercera iteración que involucran los reportes que brinda la aplicación. Al culminar esta, se consta de un producto listo para su puesta en funcionamiento.

Historia de Usuario	Estimación	Real
Mostrar Reporte General.	2	2

Tabla 20: Historia de Usuarios abordada en la tercera iteración.

4.4.1 Tareas de Historia de Usuario Abordadas en la iteración.

En las siguientes tablas han sido organizados los datos referentes a las Tareas de Ingeniería de la tercera iteración.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 5
Nombre Tarea: Creación de la interfaz Reporte General.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 22 de marzo 2009.	**Fecha Fin:** 29 de marzo 2009.
Programador Responsable: Diovis Robinet Morales, Yerandi Bracero Blanca.	

Descripción: Se crea la interfaz de reporte general.

Tabla 21: Tarea de Ingeniería Nro. 1 para la Historia de Usuario Mostrar Reporte General.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 5
Nombre Tarea: Mostrar reporte general con todos los datos de la prueba.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 29 de marzo 2009.	**Fecha Fin:** 4 de abril 2009.
Programador Responsable: Diovis Robinet Morales, Yerandi Bracero Blanca	
Descripción: Se muestra al operador en un reporte de todo lo ocurrido en el servidor de la SoftPBX Asterisk.	

Tabla 22: Tarea de Ingeniería Nro. 2 para la Historia de Usuario Mostrar Reporte General

4.5 Prueba.

Entre las prácticas de XP está la de seguir un desarrollo guiado por pruebas, posibilitando darle al cliente una idea de las verdaderas funcionalidades que tiene el producto. Mediante esta filosofía se reduce el número de errores no detectados, así como el tiempo entre la introducción de este en el sistema y su detección [16]. Realizándole al producto pruebas constantemente al final de cada iteración se eleva la calidad del mismo.

XP divide las pruebas en dos grupos:

Pruebas unitarias: Desarrolladas por los programadores y encargadas de verificar el código de forma automática.

Pruebas de aceptación: Destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

Por cada iteración se traducen las Historias de Usuarios en pruebas de aceptación, con el objetivo de demostrar al cliente que el producto cumple con los requerimientos necesarios.

Caso de Prueba de Aceptación	
Código N: HU1P1	**Número Historia de Usuario:** 1
Nombre de la Prueba: Enviar los datos referentes para la creación del Script Cliente.	
Descripción de la Prueba: Probar que se envíen correctamente los datos para la configuración del script cliente.	
Condiciones de Ejecución: El script cliente es ejecutado con privilegios de administración.	
Entrada / Pasos de ejecución: Se intenta enviar los datos desde la aplicación a los scripts clientes.	
Resultado Esperado: Los datos son enviados correctamente a los scripts clientes desde la aplicación. Lográndose una buena configuración de los scripts clientes.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 23: Caso de Prueba de Aceptación HU1P1.

Caso de Prueba de Aceptación	
Código N: HU2P1	**Número Historia de Usuario:** 2
Nombre de la Prueba: Enviar los datos referentes para la creación del Script Servidor.	
Descripción de la Prueba: Probar que se envíen correctamente los datos para la configuración del script servidor.	
Condiciones de Ejecución: El script servidor es ejecutado con privilegios de administración.	
Entrada / Pasos de ejecución: Se intenta enviar los datos desde la aplicación al script servidor después de haber configurado los scripts clientes.	
Resultado Esperado: Los datos son enviados correctamente al script servidor desde la aplicación. Lográndose una buena configuración al script servidor.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 24: Caso de Prueba de Aceptación HU2P1.

Caso de Prueba de Aceptación	
Código N: HU3P1	**Número Historia de Usuario:** 3
Nombre de la Prueba: Enviar los datos referentes para generar las llamadas.	
Descripción de la Prueba: Probar que se envíen correctamente los datos para la configuración de los scripts clientes para que comiencen a realizar las llamadas contra el servidor de la SoftPBX Asterisk.	
Condiciones de Ejecución: El script servidor es ejecutado con privilegios de administración.	
Entrada / Pasos de ejecución: Se intenta enviar los datos desde la aplicación a los script clientes para que comiencen a generar las llamadas.	
Resultado Esperado: Los datos son enviados correctamente a los script clientes desde la aplicación. En lo cuales se pudieron generar las llamadas contra el servidor de la SoftPBX Asterisk.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 25: Caso de Prueba de Aceptación HU3P1.

Caso de Prueba de Aceptación	
Código N: HU4P1	**Número Historia de Usuario:** 4
Nombre de la Prueba: Comprobar que se obtienen los datos referente a la prueba.	
Descripción de la Prueba: Probar que se obtiene correctamente los datos para mostrarlo en el reporte general.	
Condiciones de Ejecución: La aplicación es ejecutada y los datos obtenidos deben ser mostrados correctamente por la aplicación.	
Entrada / Pasos de ejecución: Se intenta obtener el reporte general.	
Resultado Esperado: Los datos referentes a la prueba son obtenidos correctamente en la aplicación**.**	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 26: Caso de Prueba de Aceptación HU4P1.

Caso de Prueba de Aceptación	
Código N: HU5P1	**Número Historia de Usuario:** 5
Nombre de la Prueba: Comprobar el reporte general de los datos de la prueba.	
Descripción de la Prueba: Probar que se muestran correctamente los datos del reporte general.	
Condiciones de Ejecución: La aplicación es ejecutada y los datos obtenidos deben ser mostrados correctamente por la aplicación.	
Entrada / Pasos de ejecución: Se intenta mostrar el reporte general.	
Resultado Esperado: Los datos del reporte son mostrados correctamente en la aplicación**.**	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 27: Caso de Prueba de Aceptación HU5P1.

4.6 Conclusiones

En el presente capítulo se abordaron las etapas de implementación y pruebas del software en desarrollo. Para ello se exponen todos los artefactos generados, realizando una descripción de cada uno de ellos.

CAPÍTULO 5: Estudio de Factibilidad.

5.1 Introducción.

En el presente capítulo se hace un análisis del costo y beneficios que tendría la realización del proyecto utilizando la planificación basada en COCOMO II (*Constructive Cost Model*) para realizar el estudio de factibilidad del producto, consiste básicamente en la aplicación de ecuaciones matemáticas sobre los Puntos de Función sin ajustar o la cantidad de líneas de código (SLOC, *Source Lines Oí Code*) estimados para un proyecto [17].

5.2 Características del proyecto.

Nombre de la entrada externa	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
Configurar Script Cliente.	1	5	Media
Configurar Script Servidor.	1	5	Media
Total		**10**	

Tabla 28: Entradas Externas.

Nombre de la salida externa	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
Mostrar Reporte General.	1	17	Media
Total		**17**	

Tabla 29: Salidas Externas.

Nombre de Fichero interno	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
Generar llamadas.	1	5	Media
Total		**5**	

Tabla 30: Ficheros Internos.

Nombre de la petición	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)

Obtener información de Prueba.	1	17	Media
Total		**17**	

Tabla 31: Peticiones.

Nombre de la interfaz externa	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
Total	**0**	**0**	

Tabla 32: Interfaces externas.

Elementos	Simple		Medio		Complejo		Subtotal
	No.	Peso	No.	Peso	No.	Peso	
Entradas Externas.	0	3	10	4	0	6	40
Salidas externas.	0	4	17	5	0	7	75
Ficheros internos.	0	7	5	10	0	15	50
Peticiones.	0	3	17	4	0	6	68
Interfaces externas.	0	5	0	7	0	10	0
Total							301

Tabla 33: Puntos de función desajustados.

5.3 Cálculo de instrucciones fuentes, esfuerzo, tiempo de desarrollo, cantidad de hombres y costo.

Características	Valor
Puntos de función desajustados.	301
Lenguaje (Python).	13
Instrucciones fuentes por puntos de función.	3913
Instrucciones fuentes.	3.91

Tabla 34: Características.

Nombre	Valor	Justificación
PREC	1.24	El sistema es muy familiar.
FLEX	2.03	Acuerdo general de flexibilidad de desarrollo.
RESL	1.41	Plan identifica la mayoría de los riesgos críticos y establece hitos para resolverlos.
TEAM	1.10	Interacciones altamente cooperativas, objetivos y culturas de accionistas fuertemente.
PMAT	4.68	Existe poca experiencia previa en el desarrollo de estos productos. La madurez del proceso de software es baja.
Total(SF)	**10.46**	

Tabla 35: Factores de escala.

Nombre	Valor	Justificación
RCPX	1.10	El sistema presenta un nivel alto de complejidad.
RUSE	1	Se pretende reutilizar el código para futuros productos en el proyecto.
PDIF	1	Uso de memoria y almacenamiento normal, plataforma estable.
PREX	1.33	Poca experiencia en el desarrollo sobre la plataforma y el lenguaje.
PERS	0.88	Alta capacidad del personal.
FCIL	0.82	Se utilizan entornos de desarrollo integrados y herramientas de modelación que facilitan el trabajo.
FCED	1.00	Se empleó el tiempo planificado para el desarrollo de la aplicación.
Total(EM)	**1.05**	

Tabla 36: Multiplicadores de esfuerzo.

Cálculos

A: se toma el valor por defecto del modelo, ajustado en 2.94

A= 2.94 B= 1.01 C= 3.67 D= 0.24 SF= 10.46 EM= 1.05

B = 0.91+0.01*SF = 0.91+0.01*10.46 = 1.01

E= B+0.01*SF = 1.01+0.01*10.46 =1.1146

PM= A*Size^E*EM = 2.94*3.91^1.1146*1.05=14.11

F= D+0.2*0.01*ΣSF = 0.24+0.2*0.01*10.46 = 0.24

TDEV= C*PM^F = 3.67*14.11^0.24=2.13 tiempo de desarrollo de la aplicación

CH= PM / TDEV = 14.11 / 2.13 = 6.62

C= CH* Sal *PM = 6.62*150*14.11 = 14011.23

Cálculo:	**Valor**
Esfuerzo.	14.11 Hombres/mes
Tiempo de desarrollo.	2.13 meses
Cantidad de hombres.	2 hombres
Salario medio.	$ 150
Costo.	$14011.23

Tabla 37: Resultados.

5.4 Beneficios tangibles e intangibles.

El beneficio fundamental del sistema es contar con una aplicación flexible, dinámica y de interfaz agradable que permita conocer de una forma más precisa y en el menor tiempo posible la información referente al rendimiento del servidor, que sirva para la toma de decisiones en la implantación de un sistema telefónico basado en Asterisk.

La aplicación no está concebida para ser un producto comercial, aunque en un futuro pueda reportar beneficios monetarios adquiriendo este carácter, por la posible venta de la misma a otras entidades para su posterior uso.

5.5 Análisis de costo y beneficios.

El desarrollo de la aplicación no constituye un gasto considerable pues todas las herramientas y tecnologías que se han empleado en su desarrollo son libres. El sistema que se propone en este trabajo no conlleva a grandes gastos, puesto que solo es influyente el salario de los desarrolladores.

Una vez implantada la herramienta le brindará al operador un sistema de prueba que permita obtener información relevante sobre el comportamiento de su servidor en condiciones de sobrecarga de procesamiento.

Teniendo en cuenta el análisis realizado se concluye que es factible la implementación de la propuesta del sistema.

5.5 Conclusiones.

En este capítulo se describió el estudio de factibilidad realizado correspondiente al sistema propuesto, teniendo en cuenta el costo estimado y los beneficios que reportará al ser implantado. El estudio realizado ha proporcionado valiosos argumentos, que permiten llegar a la conclusión de que la solución de software es factible para su puesta en funcionamiento.

CONCLUSIONES

Durante la realización de la investigación se arribaron a las siguientes conclusiones que a continuación se mencionan:

- ✓ Se logró desarrollar una herramienta capaz de medir el rendimiento de la SoftPBX Asterisk bajo condiciones de sobrecarga de procesamiento, cumpliéndose satisfactoriamente los objetivos propuestos. Incluyéndose una serie de recomendaciones que deberían tenerse en cuenta para el trabajo futuro.
- ✓ Se logró comprobar que la herramienta de prueba le permite al operador obtener información relevante sobre el comportamiento de la SoftPBX Asterisk para la toma de decisiones en la implantación de un sistema telefónico en un determinado ambiente de trabajo.
- ✓ El sistema se desarrolló siguiendo la metodología de desarrollo XP permitiendo la modelación del sistema en el tiempo estimado posibilitando así las entregas en tiempo al cliente.
- ✓ El estudio de la factibilidad determinó que el desarrollo de la aplicación es realmente factible para su puesta en funcionamiento.

RECOMENDACIONES

Una vez concluida la investigación, se realizan las siguientes recomendaciones:

- ✓ Agregar la funcionalidad de almacenar en una base de datos los resultados de las pruebas, para posteriormente poder analizarlos.
- ✓ Agregar la funcionalidad de medir la calidad de voz de cada llamada en condiciones de sobrecarga del servidor.

REFERENCIA BIBLIOGRÁFICA

1. **Página oficial del proyecto Asterisk.** Disponible en: [www.asterisk.org]
2. **Davidson, Jonatha. Peters, James.** *Voice Over IP*: Fundamentals,Tercera Edición Estados Unidos de Norte América USA: Editorial Cisco Company,2002,pp.275.Library of Congress Cataloging-in-Publication Number:99.61716
3. **Elegir un códec de audio para Asterisk.** Disponible en: [http://bytecoders.homelinux.com/content/elegir-un-c%C3%B3dec-de-audio-para-asterisk.html]
4. **Cómo escoger su proveedor de Asterisk** .Disponible en: [http://www.alfredcertain.com/?p=29/]
5. **Página oficial de SIPp**. Disponible en: [http://sipp.sourceforge.net/].
6. **Herramienta Astertest**. Disponible en: [http://www.asteriskguru.com/tutorials/astertest.html].
7. **Dimensionamiento de Asterisk**, Disponible en: [http://www.voip-info.org/wiki/view/Asterisk+dimensioning].
8. **Álvarez Angel, Miguel.** "*¿Qué es Python?"*. Disponible en: [http://www.desarrolloweb.com/articulos/1325.php].
9. **Dr. Diego Lz. de Ipiña Gz. de Artaza**. "*Python: descubre el poder del lenguaje scripting de moda en la comunidad Open Source"*. Disponible en: [http://www.e-ghost.deusto.es/docs/python.pdf].
10. **Cerón, Manuel.** *"Herramientas para Python",* Disponible en: [http://ceronman.blogspot.com/2005/12/herramientas-para-python.html].
11. **Sitio oficial de WxGlade.** Disponible en: [http://wxglade.sourceforge.net/].
12. **Catalani, Exequiel**. "*Arquitectura Modelo Vista Controlador*". Disponible en: [http://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/].
13. **Ivar, Jacobson. Grady, Booch. James, Rumbaugh**. "*El proceso unificado de desarrollo de software*". Madrid México, Editorial Addison Wesley Pearson Education.S.A .1999., 464p.ISBN:84-7829-036-2. Disponible en: [http://bibliodoc.uci.cu/pdf/reg00060.pdf]
14. **H, José. Canós, Letelier, Patricio. Penadés, Mª Carmen.** "*Metodologías Ágiles en el Desarrollo de Software*". DSIC -Universidad Politécnica de Valencia Camino de Vera s/n, 46022 Valencia. Disponible en: [http://www.willydev.net/descargas/prev/TodoAgil.Pdf].
15. **Beck K. Fowler M. (2000).** "*Planeando en Programación Extrema"*. Addison Wesley. Título original: Planning Extreme Programming.

16. **Crispin, L. House, T. (2002). "***Probando la Programación Extrema*". Addison Wesley. Título original: Testing Extreme Programming.
17. **Peralta, Mario**. "*Estimación del esfuerzo basada en casos de usos"*. Disponible en: [http://www.centros.itba.edu.ar/capis/rtis/rtis-6-1/estimacion-del-esfuerzo-basada-en-casos-de-usos.pdf]

BIBLIOGRAFÍA

1. **Van, Jim. Madsen, Leif. Smith, Jared. "***Asterisk The Future of Telephony". Versión 2,* Estados Unidos de America: O'Reilly Media, 2007.
2. **Davidson, Jonatha. Peters, James.** *Voice Over IP*: Fundamentals,Tercera Edición Estados Unidos de Norte América USA: Editorial Cisco Company,2002,pp.275.Library of Congress Cataloging-in-Publication Number:99.61716
3. **Gomillon, David. Dempster, Barrie.** *Build Telephony System with Asterisk*, Primera Edición, Estados Unidos de Norte América USA: Editorial Pack Publishing LTD,2005,pp.290,ISBN-1-904811-15-9
4. **Beck K. Fowler M. (2000).** "*Planeando en Programación Extrema"*. Addison Wesley. Título original: Planning Extreme Programming.
5. **Ignacio Moreno, Jose. Soto, Ignacio. Larrabeiti**, **David.** "*Protocolos de Señalización para el transporte de Voz sobre redes IP"*, Departamento de Ingeniería Telemática Universidad Carlos III de Madrid Avda Universidad 30, 28911. Disponible en :[http://www.it.uc3m.es/~jmoreno/articulos/protocolssenalizacion.pdf]
6. **Martínez, Felipe. Luis. Teran T, Wilson.** "*Manual de instalación y configuración de un servidor asterisk*" Junio de 2007. Disponible en :[http://www.uninorte.edu.co/divisiones/ingenierias/Dpto_Sistemas/lab_redes/upload/file/MANUAL%20DE%20INSTALACIon%20Y%20CONFIGURACIon%20DE%20UN%20SERVIDOR%20ASTERISK.pdf]
7. **Spencer, Mark. Allison, Mack. Rhodes, Christopher**. *"The Asterisk Handbook "* Disponible en : [http://www.digium.com/handbook-draft.pdf]
8. **Tomé**, **Dolores.** "*Voz sobre IP"*. Disponible en:[http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=219&mode=thread&order=0&thold=0&POSTNUKESID=7a4684da1db944c6a183c054fe10a29b]
9. **Poppendieck M., Poppendieck "**T.Lean *Software Development*: *An Agile Toolkit for Software Development Managers..." Addison Wesley. 2003*.
10. **Sheets, Kris. Terence, Jimmy. García, Marcelo. Saavedra, Dany. "***Asterisk en Español", Versión 1.* 2005. Disponible en : [http://itaki.net/espanol/asterisk_espanol.pdf]

11. **Protocolos de Voz sobre IP**. Idris 2008. Disponible en:[http://www.idris.com.ar/articulos/ART0002%20-%20Protocolos%20en%20VoIP.pdf]
12. **Viegas, Eduardo .Correa, Facundo**."Asterisk Desconsolado". Disponible en:[http://asterio.com.ar/resources/downloads/Asterisk_desconsolado.pdf]
13. **Sheets, Kris. Terence, Jimmy. Garcia, Marcelo. Saavedra, Dany**. "*Asterisk en Español* " Disponible en: [http://itaki.net/espanol/asterisk_espanol.pdf]
14. **Pascual, Escudero.Berthilson, Alberto. Louise. "***VoIP para el desarrollo*" .Disponible en :[http://www.wilac.net/doc/tricalcar/materiales_abril2008/PDF_es/16_es_voip_presentacion_v021.pdf]
15. **E, Goncalves Flavio**. "*Asterisk PBX Guía de la Configuración*" Janeiro/2007 ,362p, ISBN: 978-85-906904-3-6. Disponible en: [http://site.asteriskguide.com/FreeChapters123es.pdf]
16. **Ágil, A., 2006** .Disponible en: [http://www.agilealliance.org]
17. **Benck, K**. "*Extreme Programming Explained. Embrace Change". Pearson Education*, 1999. p. Traducido al español como: "*Una explicación de la programación extrema. Aceptar el cambio*", Addison Wesley, 2000.
18. **Crispin, L. House, T. (2002). "***Probando la Programación Extrema*". Addison Wesley. Título original: Testing Extreme Programming.
19. **Peralta, Mario.** "*Estimación del esfuerzo basada en casos de usos"*. Disponible en: [http://www.centros.itba.edu.ar/capis/rtis/rtis-6-1/estimacion-del-esfuerzo-basada-en-casos-de-usos.pdf]
20. **Catalani, Exequiel**. "*Arquitectura Modelo Vista Controlador*". Disponible en: [http://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/].
21. **Ivar, Jacobson. Grady, Booch .James, Rumbaugh.** "*El proceso unificado de desarrollo de software*". Madrid México, Editorial Addison Wesley Pearson Education.S.A .1999., 464p.ISBN:84-7829-036-2. Disponible en: [http://bibliodoc.uci.cu/pdf/reg00060.pdf]
22. **Dr. Diego Lz. de Ipiña Gz. de Artaza**. "*Python: descubre el poder del lenguaje scripting de moda en la comunidad Open Source"*. Disponible en: [http://www.e-ghost.deusto.es/docs/python.pdf].
23. **Rodríguez, Malay.** "*Introducción de procedimientos ágiles en la producción de software en la Facultad 7…".* Universidad de las Ciencias Informáticas, Ciudad de La Habana Junio 2007. 99 p Disponible en: [http://bibliodoc.uci.cu/TD/TD_0693_07.pdf]

24. **Suárez, Adilenys. Dorvigny, Darvis**. "*Redes de Voz sobre IP mediante PBX basado en Software.*" Universidad de las Ciencias Informáticas. Ciudad de La Habana Junio 2008. 70pp.
25. **Cerón, Manuel.** *"Herramientas para Python",* Disponible en: [http://ceronman.blogspot.com/2005/12/herramientas-para-python.html].
26. **Página oficial de WxGlade.** Disponible en: [http://wxglade.sourceforge.net/].
27. **Cómo escoger su proveedor de Asterisk** .Disponible en: [http://www.alfredcertain.com/?p=29/]
28. **Página oficial de SIPp**. Disponible en: [http://sipp.sourceforge.net/].
29. **Herramienta Astertest**. Disponible en: [http://www.asteriskguru.com/tutorials/astertest.html].
30. **Dimensionamiento de Asterisk**, Disponible en: [http://www.voip-info.org/wiki/view/Asterisk+dimensioning].
31. **Álvarez, Miguel.** "*¿Qué es Python?"*. Disponible en: [http://www.desarrolloweb.com/articulos/1325.php].
32. **Elegir un códec de audio para Asterisk.** Disponible en: [http://bytecoders.homelinux.com/content/elegir-un-c%C3%B3dec-de-audio-para-asterisk.html]
33. **H, José. Canós, Letelier, Patricio. Penadés, Mª Carmen.** "*Metodologías Ágiles en el Desarrollo de Software*". DSIC -Universidad Politécnica de Valencia Camino de Vera s/n, 46022 Valencia. Disponible en: [http://www.willydev.net/descargas/prev/TodoAgil.Pdf].
34. **Página oficial de la Comunidad de Usuarios de Asterisk en español**. Disponible en: [www.asterisk.es]
35. **Página oficial del proyecto Asterisk.** Disponible en: [www.asterisk.org]
36. **Página oficial VoIP**. Disponible en: [www.voip-info.org]
37. **Página oficial de Python.** Disponible en: [http://www.python.org/]

ANEXOS

Anexo I: Diagrama y descripción de las clases.

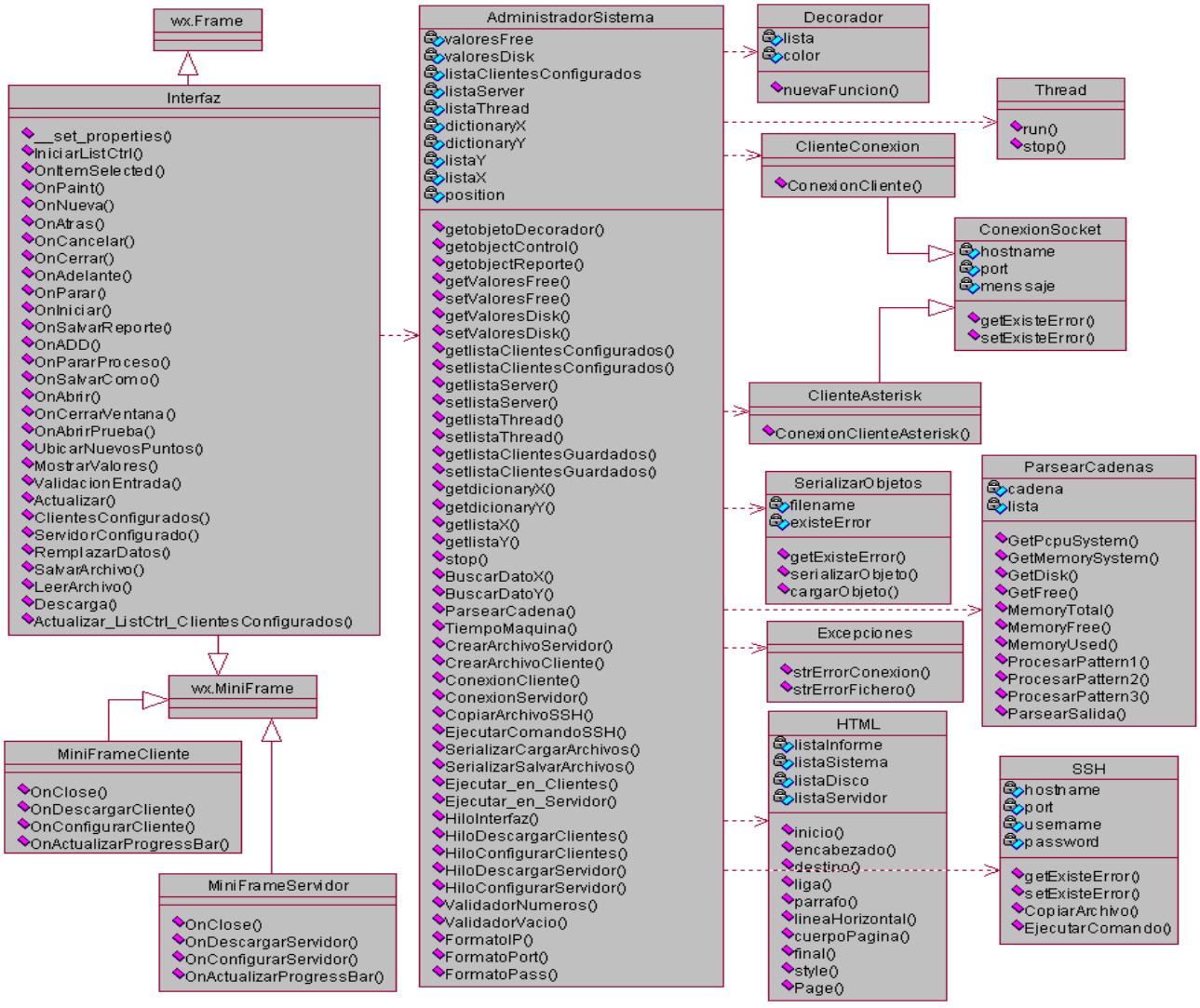


Figura 3: Diagrama de Clases.

Clases	Descripción
SSH	Clase encargada de la comunicación entre la aplicación y los servidores mediante SSH.
ConectionSocket	Clase encargada de brindar servicios de comunicación entre la aplicación y los servidores mediante la filosofía de los socket.
ClienteAsterisk	Clase encargada de brindar servicios de comunicación entre la aplicación y los servidores mediante la filosofía de los socket.
ClienteConexion	Clase encargada de brindar servicios de comunicación entre la aplicación y los servidores clientes mediante la filosofía de los socket.
Decorador	Clase encargada de brindar funciones decoradoras para validar la entrada de datos.
SerializarObjetos	Clase encargada de brindar servicios de conversión a binario.
Excepciones	Clase encargada de manejar las excepciones y errores que lance la aplicación.
ParsearCadenas	Clase encargada de tratar las peticiones realizadas al servidor a testear y obtener los valores que se desean del mismo.
AdministradorSistema	Clase encargada de administrar la lógica del sistema y la comunicación entre los modelos de clases y la Interfaz Visual.
Interfaz	Clase encargada del desarrollo de los componentes necesarios para el sistema y la interacción de estos con el Usuario.
Wx.Frame	Clase encargada de brindar una interfaz para la vista del Usuario.
Wx.MiniFrame	Clase que hereda de Wx.Frame y brinda una interfaz amigable para la vista del Usuario.
MyMiniFrameServidor	Clase encargada de brindar servicios de configuración y descarga en el servidor a testear.
MyMiniFrameCliente	Clase encargada de brindar servicios de configuración y descarga en los clientes.
Thread	Clase que trabaja a través del módulo threading y que hace posible el trabajo con hilos propios de ejecución.
HTML	Clase encargada de confeccionar el reporte en formato HTML.

Tabla 38: Descripción de las clases.

Anexo II: Diagrama de Componente.

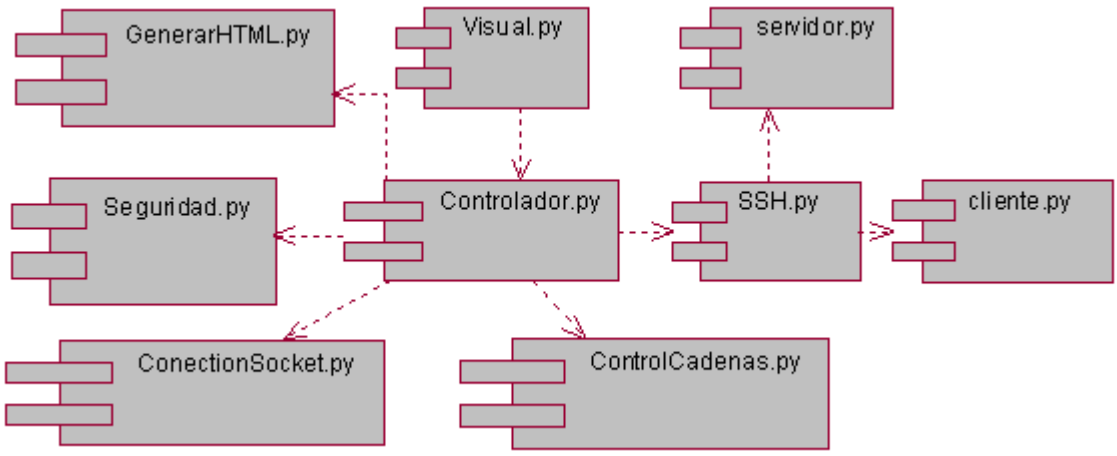


Figura 4: Diagrama de Componente.

GLOSARIO DE TÉRMINOS

Ancho de banda: Es la cantidad 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), o cualquiera de los multiplicadores. El ancho de banda a menudo se utiliza como sinónimo de la tasa de transferencia de datos.

Códecs: Es una abreviatura de codificador/decodificador, describe una especificación desarrollada en software, hardware o una combinación de ambos capaz de transformar un archivo con un flujo de datos (stream) o una señal.

Cliente: Sistema que establece un intercambio de datos con un servidor.

Digium: Empresa encargada de proveer de todo el hardware y software de Asterisk.

FSF:(acrónimo de *Fundación de Software Libre):* Organización creada con el propósito de difundir este movimiento. Está dedicada a eliminar las restricciones sobre la copia, redistribución, entendimiento y modificación de programas de computadoras.

GPL: Licencia Pública General es una licencia creada por la FSF y orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es Software Libre.

GUI: Interfaz gráfica de Usuario, componente de una aplicación informática que el Usuario visualiza y a través de la cual opera con ella. Está formada por ventanas, botones, menús e íconos, entre otros elementos. Entorno que muestra la información en pantalla en forma gráfica.

H.323: Es un standard aprobado ITU que define cómo se transmiten los datos en conferencias audiovisuales a lo largo de una red.

IDE: Es un entorno de desarrollo integrado, programa compuesto por un conjunto de herramientas para un programador desde el que se pueden editar programas, compilarlos y depurarlos.

Kernel: Núcleo o parte esencial de un sistema operativo que provee los servicios más básicos del sistema. Se encarga de gestionar los recursos como el acceso seguro al hardware de la computadora.

Libre: O software libre se refiere a la libertad de los Usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.

Metodologías: Se refiere a los métodos de investigación en una ciencia. Se entiende como la parte del proceso de investigación que permite sistematizar los métodos y las técnicas necesarios para llevarla a cabo. Define Quién debe hacer, Qué, Cuándo y Cómo debe hacerlo.

Metodologías de desarrollo: Se define como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información.

Metodología Ágil: Constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración .

MVC: (acrónimo de *Model View Controller*, Modelo Vista Controlador) es un patrón de arquitectura de software que permite realizar la programación multicapa, separando en una aplicación los datos, la interfaz del usuario y la lógica.

PBX: (acrónimo de *Private Branch Exchange,* Centralita Telefónica Privadas) interconecta las extensiones internas de los teléfonos y proporciona conexión con la red telefónica exterior.

Protocolo: Conjunto de normas que rigen un determinado proceso de comunicación.

RAM: Memoria de acceso aleatorio. Tipo de memoria donde la computadora guarda información para que pueda ser procesada más rápidamente. En la memoria RAM se almacena toda información que está siendo usada en el momento.

Routers*:* es un dispositivo de hardware para interconexión de red de computadoras que opera en la capa tres (nivel de red). Este dispositivo permite asegurar el enrutamiento de paquetes entre redes o determinar la ruta que debe tomar el paquete de datos.

RTP *(*acrónimo de *Real-time Transport Protocol):* Protocolo de transporte en tiempo real. Protocolo utilizado para la transmisión de información en tiempo real como por ejemplo audio y video en una videoconferencia.

Servidor: computadora central de un sistema de red que provee servicios y recursos (programas, comunicaciones, archivos…) a otras computadoras (clientes) conectadas a ella.

SIP: (acrónimo de *Session Initial Protocol,* Protocolo de Inicio de Sesión. Es un protocolo estándar para la iniciación, modificación y finalización de sesiones interactivas de usuarios donde intervienen elementos multimedia como el video, voz, mensajería instantánea, juegos online y realidad virtual. Protocolo para

iniciar sesiones interactivas que necesiten elementos multimedia como video, voz, chat, juegos o realidad virtual.

SIPP: Software usado en la propuesta planteada para realizar pruebas de señalización de un servidor de la SoftPBX Asterisk.

SoftPBX: PBX basado en Software; esta aplicación realiza las mismas funcionalidades de una PBX convencional.

VoIP: (acrónimo de *Voice over Internet Protocol,* Voz sobre Protocolo de Internet). Es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando un protocolo IP (Internet Protocol).

Widgets*:* Pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de widgets o Widget Engine.

XP: (acrónimo de EXtreme Programming).Metodología de desarrollo de software basada en valores como simplicidad, comunicación, retroalimentación y coraje. Es una disciplina de desarrollo de software que persigue simplificar los procesos de desarrollo. Fue diseñada para ser usada con equipos de desarrollo pequeños que necesiten desarrollos ágiles y con requerimientos cambiantes.