Universidad de las Ciencias Informáticas
La Habana. Facultad 6

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

Título: Diseño e Implementación de un Adaptador de Núcleo para el Manejador de Clúster del Servidor del Streaming Distribuido **ALLFRY´S**.

Autor: Anyer Gámez Guedes

Tutor: Ing. Yoandrys S. Pacheco Jerez

La Habana, 24 de junio de 2011
"Año 53 de la Revolución"

"El hombre ordinario solo se cuida de pasar el tiempo; el hombre de talento, de emplearlo"

Arthur Schopenhauer.

Dedicatoria

A mi querida mamá por haberme apoyado en todo momento, por confiar en mí y darme seguridad cuando asoma la desesperanza, por ser mi guía en todo momento, por todo el amor y cariño que me ha brindado, por ser tan especial en mi vida. Por darme todo y ser todo para mí.

A mí querido papá Robert por ser tan especial, por confiar en mí en todo momento y brindarme su confianza y cariño.

A mis hermanos, por ser tan importantes en mi vida y verme brindado siempre su amor y cariño.

A mis sobrinas bellas, Adriana, Alejandra y Arlet por ser tan hermosas y llenar de alegría mi corazón.

A mi abuela Josefa por ser tan especial.

A toda mi familia que adoro y está a mi lado en las buenas y las malas.

Agradecimientos.

A Dios por darme la vida y permitirme tener una familia tan maravillosa.

A la Revolución y a Fidel por haberme dado la oportunidad de estudiar en esta Universidad de Excelencia y hacer posible mi formación como ingeniero informático.

A mi mamá Noida y mi papá Robert por darme todo el apoyo y las ganas de seguir adelante, por ser mi inspiración en todo momento y ser mis guías en la vida, por ser las personas más grandes de este mundo, por no dejarme caer nunca y por estar siempre ahí para mí.

A mi tutor Yoandrys Pacheco por todo el apoyo y dedicación a este trabajo, por todos los señalamientos y recomendaciones, ha sido un honor trabajar con él.

A mis hermanos por todo su apoyo, amor y cariño y por ser tan especiales para mí.

A mis sobrinos que son los niños más lindos del mundo.

A mi abuelita Josefa por siempre orar por mí y por todo su amor y cariño.

A mi tía Nury por ser la tía más linda del mundo, por apoyarme y aconsejarme siempre.

A Ana y Molinet por ayudarme siempre que los necesite, por ser tan buenos y brindarme su apoyo.

A mis tíos Juan, Noel, Tony, Nory, Nancy por ser tan especiales y por todas sus atenciones.

A los profesores miembros del tribunal de tesis por sus consejos y recomendaciones.

A mi amigo José que es como un hermano para mí por apoyarme siempre, aconsejarme y por ayudarme siempre que necesité de él.

Agradezco a toda mi familia por brindarme su apoyo y siempre confiar en mí.

Agradecimientos

A mis compañeros de proyecto, en especial al quinteto del Server de Streaming Distribuido Leo, Frank, Leydis, Yasiel y El Rene por haber sido mis compañeros de batalla en todo momento, por ayudarme a combatir con el jefe de proyecto y obtener las victorias.

A los profesores Solangel, Víctor Frank, Jean Michel y Rafael Lorente por brindarme su apoyo y ayudarme siempre que necesite de ellos.

A Adnan por ayudarme siempre que necesité de él.

A mis compañeros inseparables a lo largo de estos 5 años: Pedro, Jose, Leo, Rafael Simón, por haber estado siempre a mi lado y apoyarme en todo, y a todos aquellos que estuvieron a mi lado y me apoyaron en el transcurso de este largo camino.

A Grechin y Sol por compartir conmigo todos estos años, por su amor y cariño.

A mis compañeros de apartamento, a Figuera, Liuver, Nilo, Humberto, Osvaldo, Rafael Simón, Frank por brindarme siempre su apoyo.

A todos mis amigos de infancia que no se encuentran aquí conmigo, pero que siempre han estado en mis pensamientos, en especial a Leandro, Víctor y Yonito.

Agradezco a todas las personas que de una forma u otra me han ayudado en la vida, me han enseñado el camino correcto y han contribuidos en mi formación profesional.

A todos ustedes infinitas gracias,

Anyer.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al centro de GEYSED de la Universidad de las Ciencias Informáticas para que hagan el uso que estimen pertinente con este trabajo.

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

_____	_____
Firma	Firma
Ing. Yoandrys S Pacheco Jerez	Anyer Gámez Guedes

TUTOR: Ing. Yoandrys S Pacheco Jerez ***(ypachecoj@uci.cu)***

Profesor instructor graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) curso 2007. Imparte la asignatura de Práctica Profesional 2 y Física. Actualmente se desempeña como arquitecto del proyecto Descomtec del centro de GEYSED de la facultad 6.

RESUMEN

La Universidad de las Ciencias Informáticas (UCI), fue creada con el propósito de brindar soluciones de software para la automatización de los procesos en todos los sectores de la sociedad cubana. En el departamento señales digitales de la facultad 6 se han venido desarrollando una serie de productos orientados al trabajo con ficheros multimedia. En el centro, la transmisión de archivos multimedia por la red se realiza a través de sistemas centralizados imposibilitando a sus proyectos contar con servicios de streaming que respondan a sus necesidades, debido a las restricciones que imponen estos tipos de servidores streaming, por lo que con la implementación del Servidor de Streaming Distribuido (SSD) se mejoró la transmisión de éstos flujos de video. El objetivo de este trabajo se centró en diseñar e implementar un Adaptador de Núcleo el cual es un subcomponente del Manejador de Clúster, y dentro de sus funciones está determinar el candidato adecuado para satisfacer la petición de un usuario, aprovechando al máximo los recursos disponibles de los servidores. Este componente mejoró el proceso de selección de respuestas a peticiones en el Servidor de Streaming Distribuido ALLFRY´S; mediante el uso de técnicas heurísticas y programación distribuida. El resultado más significativo de la investigación fue dado por la implementación de un componente capaz de seleccionar el servidor idóneo para atender una determinada petición, lo que se convirtió en un aporte para el proyecto Grupo de Desarrollo de Componentes y Tecnologías.

ABSTRACT

The University of Informatics Sciences (UCI), was created to provide software solutions for automating processes in all sectors of Cuban society. The department of digital signals of faculty 6 has developed a series of products designed to work with multimedia files. In the center, the transmission of multimedia files on the network is delivered through centralized systems, these impedes projects to have streaming services according to their needs due to restrictions imposed by these types of streaming servers, for this reason with the implementation of Distributed Streaming Server (DSS) is improved the transmission of these video streams. The aim of this work were design and implement a Core Adapter, which is a subcomponent of the Cluster Manager, and within Its functions include determining the right candidate to satisfy the request of a user, maximizing available resources of the servers. This component improved the process of selection of responses to requests in Distributed Streaming Server ALLFRY'S; by use of heuristics and distributed programming. The most significant result of the research was provided by the implementation of a component able to select the best server to attend a petition; this became a contribution to the project *Development of Components and Technologies* (DesComTec).

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Desde el triunfo de la revolución, Cuba ha seguido un camino destinado a incorporar la informática y las tecnologías de la información y las comunicaciones (TIC) a la sociedad. Como parte de esta estrategia se ha desarrollado todo un proceso de informatización que tiene como objetivo elevar la eficiencia y calidad de los servicios que se brindan en varios sectores de la sociedad cubana.

La televisión es uno de los sectores en los que se viene trabajando para cumplir las metas propuestas por el gobierno y brindarles a los televidentes un mejor servicio. Es por ello que la Universidad de las Ciencias Informáticas (UCI) se encuentra inmersa en el desarrollo de *software* con calidad en el área de la televisión. Siguiendo las políticas que se han dictado en el país sobre el uso del *software* libre y la utilización del mismo en el desarrollo de las aplicaciones para el proceso de informatización, muchas de las aplicaciones que se están desarrollando o ya están en explotación utilizan *software* libre y se despliegan sobre estaciones de trabajo con sistema operativo de código abierto, aunque se puede decir que todavía existe cierta dependencia de los *software* propietarios como es el caso del sistema operativo *Windows*.

La televisión digital terrestre (TDT) es una de las más avanzadas tecnologías existentes en el mundo, ya que permite una mejor calidad de imagen y sonido. La selección de idioma, subtítulos y la eliminación de ruidos e interferencias son algunas de las cualidades que están presentes en esta nueva tecnología, también multiplica la oferta de canales disponibles así como la posibilidad de tener acceso a servicios digitales e interactivos. Aun cuando en nuestro país este tema no es candente, el cambio hacia la TDT es inevitable. Este lento proceso de cambio se debe a que el imperio mantiene un bloqueo genocida contra la nación, el cual no permite adquirir los recursos y las tecnologías necesarias para el desarrollo del país, pero no por ello impide el desarrollo del país, lo que se evidencia en los programas de informatización de la sociedad. Para llevar a cabo un cambio hacia la TDT[1] es necesario realizar un conjunto de procesos dentro de los cuales se encuentra el desarrollo de sistemas propios, para el funcionamiento adecuado de dicha tecnología. Por tal razón es que en la UCI, universidad surgida al calor de la batalla de idea, existen diversos centros para llevar a cabo la informatización del país, uno de estos es el centro de Geoinformática y Señales Digitales (GEYSED) de la facultad 6, que se encuentra desarrollando un grupo de aplicaciones de *software* para el sector de la televisión con el fin de mejorar la calidad de los servicios,

[1] **TDT: Televisión Digital Terrestre**

brindar mayores funcionalidades y multiplicar la capacidad de la emisión. Entre estas ideas se encuentra la del desarrollo del Servidor de *Streaming* Distribuido (SSD).

El uso del internet brinda múltiples facilidades, dentro de las cuales permite realizar búsquedas y compartir información, facilita la comunicación con personas de todas partes del mundo, a través de conferencias, clases a distancias, permite ver películas, series, videoconferencias, escuchar música, entre otros muchos servicios. La reproducción de contenido multimedia a través de internet es hoy en día muy usada en las redes, a tal punto que muchas empresas productoras de DVD están cerrando por la escasa demanda de estos artículos. Antes que la tecnología *streaming* surgiera en abril de 1995, la reproducción de un archivo de video por internet implicaba tener que descargar el archivo multimedia completo al disco duro de la **PC**[2] local. Como los archivos de audio y video tienden a ser muy grandes, su descarga y acceso como paquetes completos se torna una operación muy lenta, congestionando de esta manera las grandes redes. Por tales razones es que surge el *streaming,* el cual consiste en la distribución de contenidos de audio y video a través de internet tanto en directo como en diferido.

Mediante el *streaming* es posible reproducir el audio o el video sin necesidad de descargar primero la totalidad del archivo por lo que es un sistema más rápido y dinámico que no requiere que el usuario disponga de gran cantidad de espacio en el disco. Como máximo se va almacenando la información recibida en un buffer de forma que no se produzcan pequeños cortes o pausas en caso de que la conexión a internet se ponga lenta en algún momento. Para poder transferir los datos de audio o videos a través de la red a los usuarios (clientes), se necesita de un servidor donde se encuentre el fichero a reproducirse, al que se le llama servidor de *streaming.* Este servidor será el encargado de recibir y atender las peticiones de cada cliente, así como brindarle los datos de forma continua, sincronizada y en tiempo real. **(Alvarez, 2001)**

En el mundo existen muchos tipos de servidores *streaming*, entre los que se encuentran los servidores de *streaming* centralizados, ejemplo VLC, *SHOUTcast Server*, *Subsonic*, los multiformato y distribuido como el Flumotion, otros se enfocan más a entornos de redes LAN con más disponibilidad de ancho de banda, fundamentalmente *Darwing Streaming Server*, de *Apple*, que es la versión de código abierto del *Quicktime Streaming Server* y solamente permite la transmisión por el protocolo RTP, DB2 *Digital Library Video Charger*, de IBM; *Oracle Video Server*, quiza el más utilizado de todos, con arquitectura cliente/servidor, pero con limitaciones de escalabilidad. En general, el análisis en detalle de estos productos, conduce a la

[2] **Computadora Personal**

conclusión de que la mayoría presentan soluciones caras, cerradas, no escalables y no adaptables a las necesidades del centro de GEYSED. Es por esta razón que en el centro se decidió realizar la implementación de un Servidor de *Streaming* Distribuido que responda a sus necesidades.

En la implementación del SSD[3] se presenta una arquitectura distribuida por nodos o clúster, entre los componentes de dicha arquitectura se encuentra el Manejador de Clúster, el cual cuenta con el Adaptador de Núcleo que es un subcomponente de este.

Existen diferentes tipos de adaptadores, como MPEG-21 *Digital Item Adaptation* (DIA), DIA permite la adaptación genérica de medios de comunicación. Esto significa, que en el motor de la adaptación no es necesario ningún conocimiento sobre los medios de comunicación que está a punto de adaptar.

En los últimos años la transmisión de videos por la red se ha incrementado considerablemente, siendo esta de vital importancia para brindarle los servicios a los usuarios. En el centro GEYSED[4] la transmisión de flujos video a través de la red es de gran importancia y requiere de sistemas de respuestas rápidas, por lo que las altas concurrencias en las redes provocan fallos en la transmisión de estos flujos, afectando el funcionamiento adecuado de los sistemas que necesitan de los servidores de streaming. En este centro dichos flujos se transmiten a través de sistemas centralizados, como el VLC, los cuales no permiten las altas concurrencias de usuarios.

Los servidores de s*treaming* distribuidos basados en clúster permiten un mejor rendimiento, ya que se equiparan las actividades entre todos los nodos que forman el clúster reduciendo el tiempo de respuesta, incrementando la capacidad de almacenamiento al distribuir la información, permite localizar un fallo en un nodo determinado y que sus funciones pasen a ser asumidas por otro hasta que este reanude sus funcionalidades y presentan una mayor facilidad de manejo, se pueden maniobrar todos los nodos desde un solo lugar centralizado, remoto o independiente. También se puede ganar agilidad en el mantenimiento. **(Dayrel Almaguer Chávez, 2008)**

Atendiendo a los problemas antes mencionados, se arriba al **problema de investigación:** ¿Cómo realizar la selección de un nodo del clúster para satisfacer peticiones a Manejadores de Clúster?. Para lograr esto se estudian las tecnologías libres para el desarrollo de Adaptadores de Núcleos para Manejadores de Clúster en Servidores de S*treaming* Distribuidos, cumpliendo con los **objetivos de la investigación**: desarrollar un Componente Adaptador de Núcleo para Manejadores de Clúster en Servidores de

[3] **SSD: Servidor de Streaming Distribuido**
[4] **Geoinformática y Señales Digitales**

S*treaming* Distribuidos, aplicados al **campo de acción**: las características y funcionalidades que proveen las tecnologías libres para el desarrollo de Adaptadores de Núcleos en Manejadores de Clúster para Servidores de S*treaming* Distribuidos.

Para lograr los objetivos de la investigación se planteó la siguiente **idea a defender**: la implementación de un Adaptador de Núcleo para Manejadores de Clúster permitirá un uso eficiente en la optimización de peticiones en los Servidores de *Streaming* Distribuidos.

Para dar cumplimiento a los objetivos antes planteados se definieron un grupo tareas que serán efectuadas durante la investigación, estas tareas son las siguientes:

1. Identificación de los estándares nacionales e internacionales utilizados para el desarrollo de Adaptadores de Núcleos para Servidores de *Streaming* Distribuido.
2. Caracterización de los procesos relacionados con los Adaptadores de Núcleos para Servidores de *Streaming* Distribuido.
3. Descripción del estado del arte de los Adaptadores de Núcleos para Servidores de *Streaming* Distribuidos.
4. Identificación de las funcionalidades de los Adaptadores de Núcleos para Servidores de *Streaming* Distribuido.
5. Argumentación de la metodología de desarrollo de *software* a usar en el proceso.
6. Modelado del componente Adaptador de Núcleo para Servidores de *Streaming* Distribuido. Confección de los artefactos necesarios.
7. Implementación del componente Adaptador de Núcleo para Servidores de *Streaming* Distribuido.
8. Confección de los datos de pruebas. Realizar proceso de pruebas al componente.

Para la realización de estas tareas se emplearon diferentes métodos de la investigación:

Métodos Teóricos

• “Histórico - Lógico”: Este método permite determinar las características esenciales de los adaptadores de núcleos o herramientas similares, que implementen funcionalidades que pueden servir de guía en el desarrollo del componente y realizar el estudio de las normas y principios que los rigen así como los estándares que se emplean para el desarrollo de éstos.

• “Analítico - Sintético”: permite sintetizar la información necesaria para el desarrollo del Adaptador de Núcleo, ya sea de los diferentes sistemas que permitan la adaptación o los algoritmos que se emplean en la adaptación de las peticiones. Lo que posibilita la comprensión de la misma, con ideas claras y concisas.

• “Modelado”: en el desarrollo de la investigación es necesario modelar la arquitectura que se propone, es por eso que se utiliza este método para crear modelos con vistas que favorezcan la comprensión de la misma.

Métodos Empíricos

• “Observación”: Este método facilita conocer el panorama real de una situación mediante la percepción directa, con su utilización se determinaron los rasgos imprescindibles en el desarrollo del Adaptador de Núcleo. Para determinar estos rasgos se instalaron los softwares Flumotion, PFSense y Traceroute, donde se observó el funcionamiento de ellos y una serie de parámetros fundamentales que ayudaron al desarrollo de la investigación.

El documento consta de la presente introducción, cuatros capítulos, conclusiones, recomendaciones y bibliografía. El primer capítulo está dedicado a los fundamentos teóricos para el desarrollo del Adaptador de Núcleo, los conceptos asociados con los adaptadores de núcleo, Servidores de *Streaming* Distribuidos, así como el análisis de otras soluciones existentes en el mundo y su desarrollo en el país. El segundo capítulo está dedicado a la realización de una valoración crítica de las herramientas, metodologías y tecnologías que serán empleadas en el desarrollo del sistema. En el tercer capítulo se encuentra la propuesta y solución del componente, dónde se modela el sistema, se identifican las funcionalidades que presenta el componente y las cualidades que debe cumplir. El cuarto capítulo está dedicado a la implementación y pruebas, en el que se aprecian los artefactos del modelo de implementación, se muestra la estructura del componente y las diferentes pruebas realizadas al mismo para encontrar errores y probar el sistema.

1. CAPÍTULO 1: Fundamentación Teórica.

1.1. Introducción

En el presente capítulo se brinda una visión general de los aspectos teóricos relacionados con la tecnología *streaming*, los servidores *streaming,* los métodos heurísticos, programación distribuida y arquitectura cliente – servidor, se mencionan algunas tendencias actuales y sus características. Además, se realiza un estudio crítico valorativo de las diferentes herramientas existentes en el mundo que posibilitan un mejor entendimiento de las posibles funcionalidades del Adaptador de Núcleo.

1.2. Conceptos asociados al dominio del problema

A continuación se muestran los conceptos fundamentales asociados al dominio del problema. Al analizar todo el proceso de funcionamiento del Adaptador de Núcleo, se puede ver evidenciado cada uno de estos conceptos en su flujo de trabajo, por lo que se cree que para un mejor entendimiento es preciso conocer el significado de cada uno de estas actividades que intervienen en todo el flujo de actividades del componente.

1.2.1. ¿Qué es Streaming?

Es una tecnología que permite la visualización y audición de un archivo mientras se está descargando, es decir, no es necesario descargar completo el contenido multimedia para poder mostrarlo, este proceso se realiza a través de la construcción de un ***buffer***[5] por parte del cliente, una vez que se ha conectado al servidor, el ***buffer*** del cliente se va llenando de la información descargada y se va reproduciendo en el ordenador. El sistema se encuentra sincronizado, tal que, una vez terminada la reproducción del contenido del archivo, finaliza la descarga **(Marcos Guglielmetti, 2005)**. El *streaming* puede ser **bajo demanda** o **en vivo**.

Su orientación está dirigida fundamentalmente para ser usado en Internet, ya que a pesar de que puede ser reproducido desde el propio disco duro, será de mayor utilidad cuando sea reproducido en un ordenador y el contenido del archivo en cuestión se encuentre en otro ordenador a muchos kilómetros de

[5] Espacio de memoria que se utiliza como regulador y sistema de almacenamiento intermedio entre dispositivos de un sistema informático.

distancia conectado a través de una red LAN, WAN o la misma Internet. La figura 1 muestra los diferentes componentes que interactúan al realizarse el streaming.

Streaming bajo demanda: *Los contenidos multimedia almacenados en el servidor son difundidos a los usuarios tras una petición. Un usuario que pulsa sobre un enlace a un clip bajo demanda ve el clip desde el principio y tiene control sobre el mismo, puede aplicar las características de avance rápido, rebobinado, pausa y reinicio del contenido* **(Torres, 2009)**.

Emisión en vivo (*broadcast*): Los contenidos en vivo pueden ser difundidos de diferentes maneras. El administrador decidirá qué método usar en función de las necesidades de la red. *Un usuario que pulsa sobre un enlace a un contenido en vivo se conectará al evento en curso y, dado que está ocurriendo en tiempo real, no puede tener control sobre el contenido* **(Torres, 2009)**.

Los clips en vivo se emiten según son creados. Estos clips no existen como archivos, porque son creados a medida que el evento sucede. Se puede guardar contenido en vivo como archivos, los ficheros guardados se convierten en contenido bajo demanda y son tratados como tales desde ese momento.

De manera general estas son algunas de las características que definen la tecnología streaming*:*

- Se utiliza para sistemas multimedia distribuidos.
- Se emplean en la transmisión de información multimedia de tipo continuo.
- Segmentan la información para transmitirla.
- El envío de información se realiza de forma temporizada y separada por flujos.
- La reproducción puede comenzar instantes después del comienzo de la transmisión.
- No es necesario que el cliente almacene toda la información que recibe.

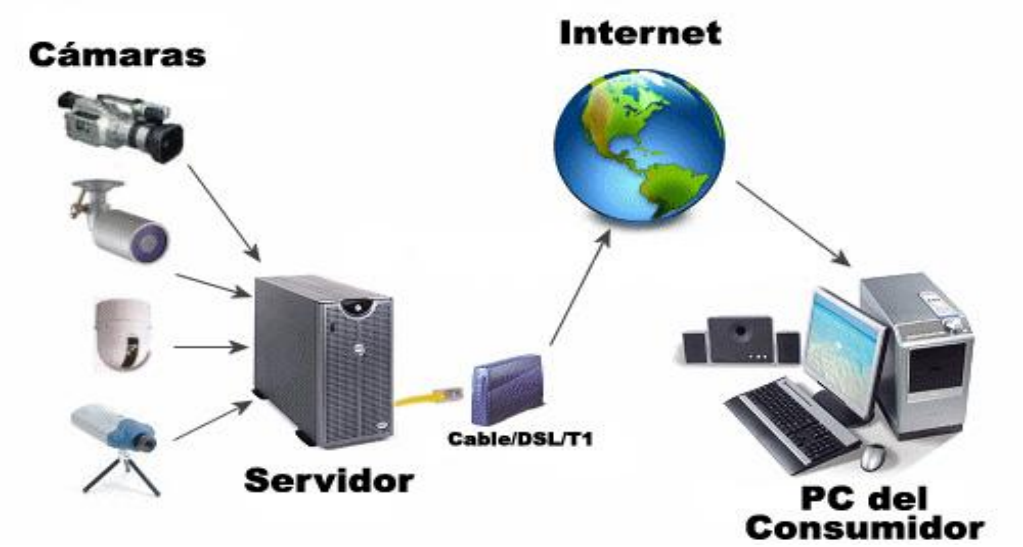

Fig. 1. Componentes de Streaming. **(Tusch, y otros)**

1.2.2. ¿Qué es un servidor de streaming?

Son programas diseñados exclusivamente para que puedan enviar múltiples flujos de información de manera continua. Estos programas, para su óptimo funcionamiento, deben ejecutarse en ordenadores conectados a internet con un ancho de banda amplio, capaz de poder atender las peticiones de los usuarios que accederán a él.

Otra definición

Según **(Fajardo, 2007),** un servidor de streaming es el conjunto de hardware con el software necesario, para cumplir lo función principal de servir audio o vídeo y proporcionar un flujo continuo de pequeños paquetes de información (audio o video) con requisitos de tiempo real. Independientemente del tipo de arquitectura.

En términos generales se puede decir que un servidor de streaming es el principal elemento en cuanto a calidad del servicio se refiere. El servidor procesa los datos multimedia en cortos espacios de tiempo y soporta funciones de control interactivas siendo además el responsable de suministrar los servicios de audio y video en modo sincronizado. El servidor de streaming espera de la petición del usuario y cuando recibe una petición, el servidor busca en el directorio apropiado el archivo solicitado. Las entregas de

paquetes al usuario pueden estar sujetas a demoras conocidas como **lag**[6]. Entonces referente a los servidores de streaming se puede decir que estará formado por el software para realizar streaming y un sistema de catalogación en caso de ser necesario.

¿Qué es un Servidor de Streaming Distribuido?

Un servidor de streaming distribuido es un servidor de streaming con la particularidad de que se distribuyen las actividades en varias computadoras con el objetivo de garantizar una mayor disponibilidad y un mejor funcionamiento.

Servidor de Streaming Distribuido atendiendo a la arquitectura ADMS

Un servidor de streaming distribuido es un servidor que cuenta con cuatro componentes básicos que son necesarios y suficientes para la composición de una dinámica reconfigurable. Estos componentes son: Manejador de Clúster, Manejador de Datos, Distribuidor de Datos y Colector de Datos. La figura 2 muestra la interrelación de los componentes. **(Tusch, y otros, 2004)**

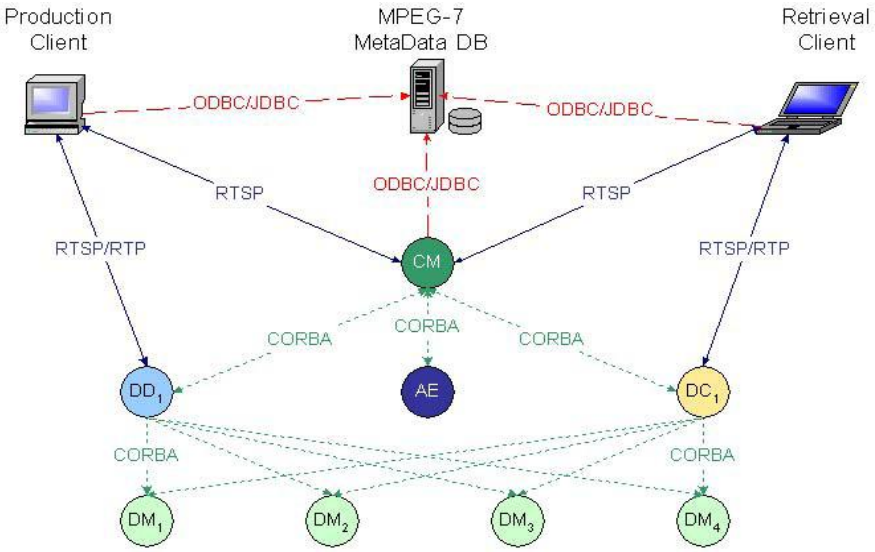

Fig.2. Componentes del servidor de streaming distribuido **(Tusch, et al., 2004)**

[6] **lag:** Transmisión a velocidades menores a la del ancho de banda donde se pierde calidad en la reproducción

Manejador de Clúster

El Manejador de Clúster es el componente central en la arquitectura ADMS (Adaptive Distributed Multimedia Server). Es nombrado director del clúster, ya que gestiona el número y la ubicación de los distribuidores datos, manejadores de datos y los colectores de datos que resulta en un clúster de servidores de aplicaciones dinámicas ADMS. Por otra parte, representa el punto central para el manejo de las conexiones de los clientes, aunque no atiende las solicitudes del cliente por sí mismo, las dirige adecuadamente a un distribuidor o colector.

Manejador de Datos

Un Manejador de Datos es el que provee de manera eficiente el almacenamiento y la recuperación de los datos en un servidor streaming, después que el Distribuidor de Datos que es otro de los componentes con que consta el servidor streaming realice la operación de separación de segmentos, el Manejador de Datos es el encargado de almacenarlo. Desde un streams elemental puede estar listado un sin número de información, cada Manejador de Datos solo almacena una porción del streams. El mismo lo organiza internamente mediante los segmentos de media almacenando, solamente guardando la información que contengan las mismas y su dirección en memoria, utilizando un set de media para cada división del segmento, formando así un fragmento de streams, lo cual permite una mayor eficiencia en el almacenamiento. A la hora de recuperar dichos segmentos se realiza una petición del Colector de Datos, el cual va a buscar donde lo tiene almacenado mediante el nombre y la dirección para así devolverlo en segmento.

Distribuidor de Datos

Es el encargado de reducir y segmentar el contenido multimedia en pequeños fragmentos.

Colector de Datos

El Colector (recopilador) de Datos es un componente del servidor streaming distribuido que realiza la operación inversa al distribuidor de datos. El mismo recolecta las unidades de bandas de un paquete de medias con la configuración apropiada del Manejador de Datos, recoge estos paquetes de diferentes orígenes de datos, después envía las unidades en secuencia de buffer hacia los clientes a través de una conexión por RTP (Protocolo de Transporte de Tiempo real). Esto proporciona tolerancia a fallos a nivel de servidor mediante la explotación de unidades de la paridad. El colector también puede incorporar un

componente de almacenamiento en caché, lo que reduce las latencias de cliente al inicio y el consumo de ancho de banda, además evita las recargas en los servidores de almacenamiento.

1.2.3. Adaptador de Núcleo

El Adaptador de Núcleo es un subcomponente del Manejador de Clúster y dentro de sus funcionalidades está la aplicación de criterios de selección para determinar dado un conjunto de nodos el candidato adecuado para satisfacer determinadas solicitudes. Se basa fundamentalmente en la cantidad de subredes por la que tiene que atravesar una petición desde donde la realizó el cliente hasta los servidores, además de la cantidad de recursos disponibles de los servidores. Utiliza los criterios de selección para aplicar un algoritmo heurístico de selección de atributos y obtener el resultado esperado.

Para obtener el resultado esperado el algoritmo evalúa una heurística, función matemática que sirve para obtener en un proceso de búsqueda el camino más corto de un nodo dado hacia un nodo objetivo, la misma se origina en dependencia de los criterios de selección de cada nodo, el algoritmo examina cada uno de los nodos usando una estrategia completa y se basa en la medida de la información que le brindan estos nodos. Este proceso se realiza bajo el paradigma de la programación distribuida con la arquitectura cliente – servidor, que está enfocada a desarrollar sistemas distribuidos, abiertos, escalables, transparentes y tolerantes a fallos, con el objetivo de maximizar la eficiencia de las respuestas. Este paradigma es el resultado del uso de las computadoras (nodos) y las redes.

1.2.4. Heurística

Heurística es la capacidad que ostenta un sistema determinado para realizar de manera inmediata innovaciones positivas para sí mismo y sus propósitos. En programación se dice que un algoritmo es heurístico cuando la solución no se determina en forma directa, sino mediante ensayos, pruebas y reensayos. Es una especie de algoritmo inteligente de búsqueda **(Bijit)**. La Figura 3 muestra las características fundamentales del concepto de heurística.

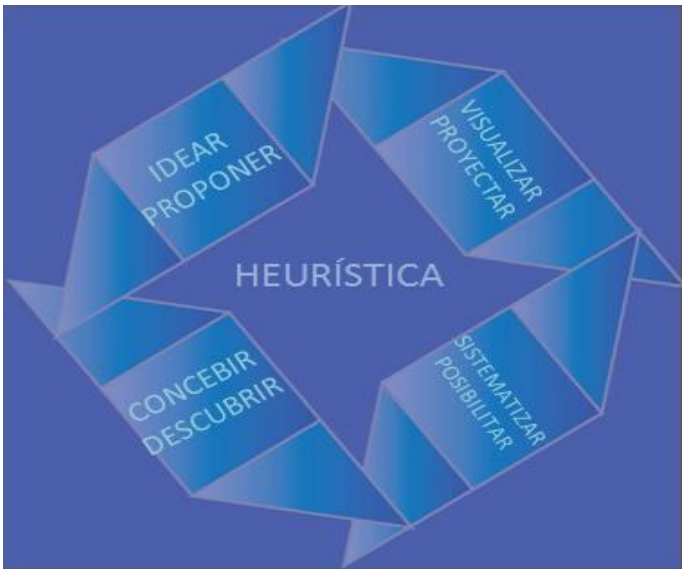

Fig.3. Definición de Heurística (Bijit)

El paradigma heurístico nace como necesidad de encontrar programas capaces de resolver los típicos problemas de búsqueda de la mejor solución cuando las alternativas son numerosas, tales como:

- Cuál es el mejor medio para transportar, desde diversos orígenes hacia diferentes destinos.
- Determinar los mejores postulantes para asignarles determinadas tareas entre varias posibles.
- Detectar dentro de las múltiples posibilidades, todos los posibles casos de cierta característica.

Para resolver este tipo de planteamientos se recurre a la programación heurística asociada a la inteligencia artificial, que es la rama de la ciencia que, simulando la operatividad de su razonamiento intenta emular en las computadoras los procesos mentales inteligentes del hombre. Con tal fin, es necesario aplicar heurísticas, que son elementos operativos que resultan de agregarles conocimiento a los métodos de búsqueda para hacerlos eficientes.

1.2.5. Métodos de búsquedas heurísticas

En el mundo existen muchos métodos de búsquedas heurísticas entre los que se encuentran el Escalador de Colina, Primero el Mejor y A*. Cada uno de estos métodos se implementa de forma diferente, teniendo en cuenta parámetros diferentes en escenarios distintos. A continuación se describen estos métodos.

Escalador de Colinas (Hill Climbing)

El Escalador de Colinas usa una técnica de mejoramiento iterativo que comienza a partir de un punto (punto actual) en el espacio de búsqueda. Si el nuevo punto es mejor se transforma en el punto actual, si no, otro punto vecino es seleccionado y evaluado. El método termina cuando no hay mejoría o cuando alcanza un número predefinido de iteraciones, puede que nunca llegue a encontrar una solución, si son atrapados en estados que no son el objetivo, desde el cuál no se puede obtener mejores estados. El escalador de colinas no es exhaustivo ya que no explora todo el espacio de estados y es eficiente pero evita la exploración de una parte del espacio de estados.

Primero el Mejor (Best-First)

Primero el Mejor combina las ventajas de los algoritmos primero en profundidad y primero en amplitud. Sigue un sendero a la vez, pero puede cambiarse a otro sendero que parece más prometedor que el que está siguiendo.

Este algoritmo tiene como características fundamentales que:

- Se tiene un control de todos los estados terminales y sus heurísticas.
- Se toma como raíz el estado de valor óptimo entre todos los estados terminales.
- Se generan en amplitud todos los estados hijos del óptimo (expandir el nodo más idóneo).

La idea del best-first es utilizar una función de evaluación para cada nodo (estimación de la idoneidad de expandir ese nodo). Aplicación: Expandir siempre el nodo no expandido más idóneo.

A* (A asterisco o A estrella)

El algoritmo A* utiliza una función de evaluación $f(n) = g(n) + h'(n)$, donde $h'(n)$ representa el valor heurístico del nodo a evaluar desde el actual, n, hasta el final, y $g(n)$, el coste real del camino recorrido para llegar a dicho nodo, n. A* mantiene dos estructuras de datos auxiliares, que se pueden denominar abiertos, implementado como una cola de prioridad (ordenada por el valor $f(n)$ de cada nodo), y *cerrados*, donde se guarda la información de los nodos que ya han sido visitados. En cada paso del algoritmo, se expande el nodo que esté primero en abiertos, y en caso de que no sea un nodo objetivo, calcula la $f(n)$ de todos sus hijos, los inserta en abiertos, y pasa el nodo evaluado a cerrados.

El algoritmo es una combinación entre búsquedas del tipo primero en anchura con primero en profundidad: mientras que h(n) tiende a primero en profundidad, g(n) tiende a primero en anchura. De este

modo, se cambia de camino de búsqueda cada vez que existen nodos más prometedores. Como todo algoritmo de búsqueda en anchura, A* es un algoritmo completo: en caso de existir una solución, siempre dará con ella.

1.2.6. Programación heurística

La programación heurística es el tipo de programación computarizada que resuelve problemas, aplicando reglas de buena lógica, denominadas heurísticas, las cuales proporcionan entre varias alternativas de acción la óptima. La programación heurística constituye una forma de representar la estructura, estrategias de búsqueda y métodos de resolución de problemas. **(Borges)**

Esta es aplicada frecuentemente en el campo de la Inteligencia Artificial, especialmente en la Ingeniería del Conocimiento, dado que el hombre opera generalmente aplicando heurísticas, que constituyen conclusiones del razonamiento humano sobre un tema específico, apoyado en la experiencia, utilizando reglas de buena lógica.

La programación heurística permite procesar por computadora los siguientes tipos de problemas:

a) ¿Cuál es el mejor camino?

b) Listar todos los casos posibles.

c) ¿Existe una asignación de elementos que satisfaga?

El común denominador de estos problemas es la necesidad de buscar exhaustivamente entre las posibles, un conjunto finito de alternativas, cuya cantidad de elementos esté controlada y de esta manera poder evitar un crecimiento exponencial del espacio de búsqueda, que sería imposible de procesar. **(Borges)**

La programación heurística se presenta como:

a) Técnica de búsqueda para obtener metas en problemas no algorítmicos, o con algoritmos que generan como en el caso del juego damas o ajedrez, múltiples alternativas de acción.

b) Método aproximado de resolución de problemas usando funciones de evaluación heurísticas.

c) Método de poda para estrategias de programas que juegan.

1.2.7. Programación distribuida

Es un paradigma de programación enfocado en desarrollar sistemas distribuidos, abiertos, escalables, transparentes y tolerantes a fallos. Este paradigma es el resultado natural del uso de las computadoras y las redes. La programación distribuida típicamente cae en alguna de las varias arquitecturas básicas: cliente-servidor, **3-tier**[7], **n-tier**[8], objetos distribuidos, entre otras; además de ser base para la pragmatisidad. Los lenguajes específicamente diseñados para programación distribuida son: Ada, Alef, E, Erlang, Limbo y Oz. **(María F. Rosique Contreras, 2003)**

Ventajas de la Programación Distribuida.

Cualquier lenguaje de programación que tenga acceso al máximo de hardware del sistema puede manejar la programación distribuida, considerando una buena cantidad de tiempo y código.

- Con respecto a Sistemas Centralizados:
  - Una de las ventajas de los sistemas distribuidos es la economía, pues es mucho más barato, añadir servidores y clientes cuando se requiere aumentar la potencia de procesamiento.
  - El trabajo en conjunto. Por ejemplo: en una fábrica de ensamblado, los robots tienen sus CPUs diferentes y realizan acciones en conjunto, dirigidos por un sistema distribuido.
  - Tienen una mayor confiabilidad. Al estar distribuida la carga de trabajo en muchas máquinas la falla de una de ellas no afecta a las demás, el sistema sobrevive como un todo.
  - Capacidad de crecimiento incremental. Se puede añadir procesadores al sistema incrementando su potencia en forma gradual según sus necesidades.
- Con respecto a PCs Independientes:
  - Se pueden compartir recursos, como programas y periféricos, muy costosos. Ejemplo: Impresora Láser, dispositivos de almacenamiento masivo, etc.
  - Al compartir recursos, satisfacen las necesidades de muchos usuarios a la vez. Ejemplo: Sistemas de reservas de aerolíneas.
  - Se logra una mejor comunicación entre las personas. Ejemplo: el correo electrónico.
  - Tienen mayor flexibilidad, la carga de trabajo se puede distribuir entre diferentes ordenadores.

[7] **Arquitectura en tres capas**
[8] **Arqiutectura en n capas**

1.2.8. Modelo Cliente – Servidor

El Modelo Cliente – Servidor según **(Durán),** es la implementación de una aplicación en capas que está basada en el envío de mensaje y representa una estructura modular que mejora la usabilidad, flexibilidad, interoperabilidad y la escalabilidad.

El modelo cliente – servidor precisa además de unos recursos (software), todos estos recursos son manejados por los procesos del servidor. Cliente y servidor deben hablar el mismo lenguaje para conseguir una comunicación efectiva, el primero solicita al segundo unos recursos y este último los concede, le hace esperar o lo deniega según los permisos que tenga. Uno o más servidores crean unos objetos locales y luego atienden peticiones de acceso sobre esos objetos proveniente de clientes situados en lugares remotos de la red. La Figura 4 es una representación del modelo cliente – servidor. **(María F. Rosique Contreras, 2003)**

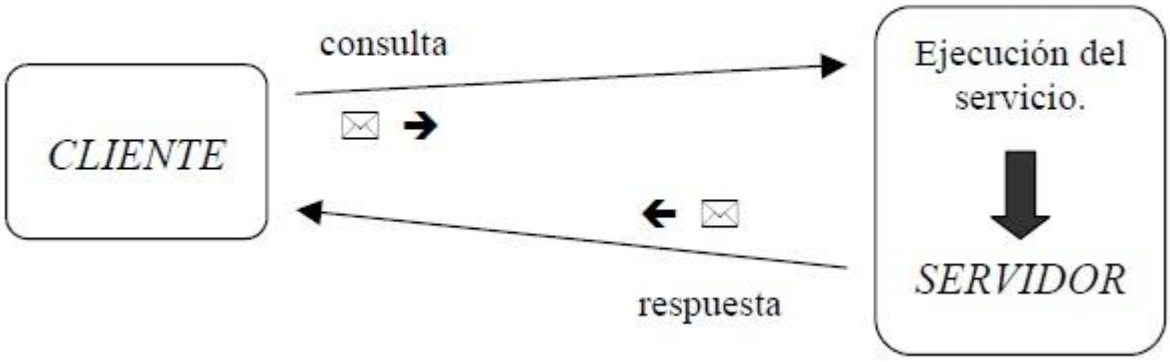

Fig.4. Modelo Cliente – Servidor (María F. Rosique Contreras, 2003)

En el modelo orientado a objetos hay una serie de objetos que solicitan servicios (clientes) a los proveedores de los servicios (servidores) a través de una interfaz de encapsulación definida. Un cliente envía un mensaje a un objeto (servidor) y este decide qué tecnología ejecutar. Para establecer esta comunicación entre los clientes y servidores existen tecnologías como CORBA, RMI, ICE, entre otras que permiten establecer una interfaz de comunicación entre aplicaciones.

1.2.9. ¿Qué es Clúster?

Un clúster es un conjunto de computadoras, a menudo con semejantes componentes de hardware, que se interconectan entre sí a través de un sistema de red de alta velocidad y son capaces de elevar la eficiencia para realizar determinadas tareas que individualmente no podrían realizar debido a la creciente necesidad de potencia computacional que demandan algunas aplicaciones **(Dayrel Almaguer Chávez, 2008)**. A pesar de estar compuesto por varias computadoras, la arquitectura de un clúster es completamente transparente para el usuario final, por lo que este lo ve como un único sistema que representa una máquina con altos niveles de procesamiento.

Ventajas que ofrece

La implementación de un clúster brinda gran flexibilidad dada por las características de sus nodos. Los nodos pueden tener las mismas configuraciones de hardware y sistema operativo, cuando esto ocurre se le llama clúster homogéneo. También puede que cada uno de ellos alcance un rendimiento diferente, proporcionado por la variedad de características de hardware, pero con arquitecturas y sistemas operativos similares, en cuyo caso se denomina clúster semihomogéneo. Como última alternativa cuando los nodos tienen diferente hardware y sistema operativo se le llama clúster heterogéneo. La flexibilidad de un clúster hace más fácil y económica su construcción.

Cada uno de los nodos de un clúster puede ser un sistema completo para usar un amplio rango de aplicaciones, por ejemplo, un nodo puede constituir un servidor de bases de datos y contener la información utilizada por diferentes aplicaciones, estos nodos se pueden reemplazar fácilmente en caso de que su funcionamiento no sea correcto e incluso se pueden integrar nuevos nodos a la granja de servidores lo que convierte al clúster en un sistema altamente escalable. Esta característica permite al sistema adecuarse en todo momento según las necesidades existentes, que pueden ser cambiantes, y hacerlo con el mínimo costo.

Además, se pueden formar sistemas verdaderamente grandes que comprenden desde dos hasta varios cientos de nodos lo que aumenta la disponibilidad de los servicios y hace que estos se brinden de forma más eficiente. El escalamiento de un clúster puede producirse de manera vertical u horizontal. Generalmente el escalar verticalmente o escalar hacia arriba, significa el añadir más recursos a un nodo en particular mientras que el escalado horizontal significa agregar más nodos al sistema.

Hoy en día las llamadas supercomputadoras son equipos excesivamente caros mientras que la implementación de un clúster resulta más barata y económica. Se pueden confeccionar utilizando simples ordenadores de escritorio que llegue a ofrecer rendimiento muy cercano al alcanzado por una supercomputadora en cuanto a poder de cómputo. Por otra parte, el hardware de red necesario para la interconexión de los nodos del clúster experimenta a medida que transcurre el tiempo un decremento constante de precio incluso se pueden lograr ahorros adicionales empleando un solo monitor, mouse y teclado para la administración de todo el sistema.

Clasificación de los clúster

De acuerdo con los servicios que prestan puede ser clasificado de tres formas:

- Clúster de alta disponibilidad.
- Clúster de alto rendimiento.
- Clúster de balanceo de carga.

Un clúster de alta disponibilidad tiene el objetivo de proporcionar máxima disponibilidad y confiabilidad de los servicios que ofrece de tal manera que estos se brinden ininterrumpidamente. Se consigue alta disponibilidad haciendo redundantes los sistemas y de esta forma cuando se produce la caída del nodo que brinda el servicio un clon del mismo puede tomar el control y comenzar a servir el servicio nuevamente. La confiabilidad se consigue mediante un software que detecta fallos y permite la recuperación frente a los mismos. Un clúster de alta disponibilidad evita que el sistema tenga un único punto vulnerable a la ocurrencia de fallos.

En un clúster de alto rendimiento se ejecutan tareas que requieren de gran capacidad computacional, grandes cantidades de memoria, o ambos a la vez. Este tipo de clúster divide las tareas en tareas más pequeñas y las reparte entre los nodos que lo conforman para que sean calculadas en ellos y así agilizar el procesamiento de los datos. Estos sistemas se implementan en un ambiente de programación paralela utilizando algoritmos que hacen uso de recursos compartidos tales como CPU, memoria, datos y servicios. Entre las aplicaciones más comunes de clúster de alto rendimiento se encuentra el pronóstico numérico del estado del tiempo, astronomía, investigación en criptografía, análisis de imágenes, entre otras.

El clúster de balanceo de carga se encarga de colocar en paralelo varios servidores (servidores reales) capaces de brindar el mismo servicio y de alguna forma repartir el trabajo entre ellos, tal que los clientes

vean servidas sus solicitudes en tiempos menores y aceptables. Los clientes deben ver al conjunto de servidores como si fuera uno solo (servidor virtual). Un clúster de balanceo de carga tiene peculiaridades de clúster de alta disponibilidad y alto rendimiento.

1.3. Análisis de tecnologías existentes

En sentido general existen diversos servidores de streaming, tanto comerciales como de código abierto y centralizado, ya que no distribuyen entre nodos las peticiones, existen otros como el Flumotion que presentan una arquitectura distribuida. En la actualidad existen aplicaciones que implementan tecnología de balanceo de carga; a nivel mundial se encuentran muchas herramientas que funcionan bajo los principios del Adaptador de Núcleo, es decir, herramientas que permiten optimizar el procesamiento de información, y a su vez responder de forma rápida y transparente. A continuación se analizan algunas de las herramientas, mediante este análisis se pretende guiar el trabajo del componente Adaptador de Núcleo, sentando pautas para el desarrollo del mismo.

1.3.1. Flumotion Streaming Server

Flumotion Streaming Server ofrece una solución de streaming multiformato, distribuida y de código abierto. Es un servidor de streaming compatible con los formatos más utilizados en Internet (flv, wmv, MP3, H264, AAC, AAC) que incluye la adquisición en su diseño descentralizado. Flumotion Streaming Server permite una gran escalabilidad y su administración es sencilla. Este software utiliza una tecnología homogénea que funciona sobre Linux. Esta tecnología es de desarrollo propio, fruto de un proceso de I+D (investigación y desarrollo) de más de dos años y medio, abarca todo el proceso de entrega de contenidos y cualquier aspecto relacionado, como la inserción de publicidad, autenticación de usuarios, soluciones de pago o gestión de derechos de emisión.

Flumotion permite repartir la carga del proceso de streaming entre diferentes nodos y no tiene limitaciones en el ancho de banda. Para poder lograr esta escalabilidad basa su funcionamiento en el ancho de banda de la red, adaptándose en todo momento al tipo de conexión de cada usuario: **"adaptive bitrate"**, de esta forma los mismos pueden ver los videos en una calidad adaptada a su conexión y dispositivo. El número máximo de usuarios concurrentes depende del bitrate al que se ha codificado el contenido, si este número máximo de usuarios concurrentes es muy amplio, entonces significa que excede el máximo ancho de banda disponible y no podrá tener más usuarios concurrentes conectados, sin embargo, los que ya se han conectado no tendrían ningún problema puesto que no verán alterada la calidad del servicio. Esto se debe

al agotamiento del ancho de banda y por ende las peticiones serán rechazadas, para poder garantizar el adecuado funcionamiento de los servicios que se le están brindados a los usuarios en ese momento.

Se puede decir que Flumotion representa un gran potencial, ya que es un servidor distribuido que implementa el balanceo de carga, basado en los recursos disponibles de los nodos servidores, pero a pesar de estar distribuido en nodos, a la hora de seleccionar el nodo idóneo para satisfacer una petición basa su funcionamiento de red fundamentalmente en el ancho de banda con el **"adaptive bitrate",** por lo que si se tiene un servidor en la misma subred del usuario que realiza la petición, se corre el riesgo de que no sea seleccionado debido a que Flumotion no tiene en cuenta el salto de las subredes, elemento que se tendrá en cuenta en el desarrollo del Adaptador de Núcleo.

1.3.2. PFSense

PFSense es una distribución Berkeley Software Distribution (BSD) basada en FreeBSD por lo que es de libre distribución, está destinado a ser instalado en un ordenador personal y ofrece características que a menudo solo se encuentran en costosos servidores de seguridad comercial.

Esta aplicación trata de repartir las peticiones que vienen desde los clientes entre todos los servidores web existentes, PFSense es como un sistema operativo, por la razón que al instalarlo se apodera de todo el disco duro y solo eso se puede instalar en una máquina.

Este software permite dividir las tareas que tendría que soportar una única máquina, con el fin de maximizar las capacidades de proceso de datos, así como de ejecución de tareas, es capaz de garantizar una alta disponibilidad y permite que una parada del servicio tenga consecuencias mínimas para los usuarios, por lo que podrían seguir trabajando de forma transparente debido a que este sistema tiene la capacidad de ofrecer el servicio asignado a las máquinas de forma continua, incluso en caso de fallo de una de ellas. Esto permite que se pueda evitar que picos de acceso a las maquinas (como por ejemplo los generados por campañas publicitarias), afecten el normal funcionamiento del aplicativo y permite gestionar y optimizar todos los recursos disponibles dando como resultado un acceso más rápido y estable a las aplicaciones.

Mediante la implementación de este sistema de balanceo de carga se optimizan las peticiones recibidas al servidor de streaming por lo que el Adaptador de Núcleo basará su funcionamiento en un modelo de balanceo de carga similar al de este software, teniendo en cuenta que el mismo permite gestionar y optimizar los recursos de los servidores, funcionalidad de vital importancia en el desarrollo del Adaptador de Núcleo.

1.3.3. Traceroute

Traceroute es una herramienta de diagnóstico de redes, presente en la mayoría de los sistemas operativos. Esta herramienta permite determinar la ruta efectuada por un paquete, funciona modificando el campo TTL en los paquetes IP. Cada paquete IP posee un campo de vida útil (TTL) el cual se reduce cada vez que pasa por un router.

Traceroute envía paquetes a un puerto UDP sin privilegios, el cual se cree que no está en uso, con un TTL configurado en 1. El primer router encontrado eliminará el paquete y enviará un mensaje ICMP que incluye la dirección IP del router y la demora del bucle. Luego, el Traceroute aumenta el campo TTL de 1 en 1, para obtener una respuesta de cada router en la ruta, hasta que obtiene la respuesta "puerto ICMP inalcanzable" de la máquina destino.

Esta herramienta es de gran utilidad y presenta características importantes, presenta muchas funcionalidades para el uso en las redes, principalmente calcular la cantidad de saltos de un paquete, el cual basa su funcionamiento en las rutas que deben seguir los paquetes que se encuentran en los routers a través de sus tablas de enrutamiento, esta aplicación permite guiar el trabajo del Adaptador de Núcleo, ya que el mismo al recibir una petición tiene que calcular la cantidad de saltos (cantidad de subredes) que existen desde el host del cliente hasta los servidores en cuestión, para de esta forma aplicar el algoritmo de selección y poder satisfacer la petición de manera correcta. Este cálculo se realiza desde cada uno de los servidores hasta el host cliente para determinar la ruta más corta, parámetro que influye de manera significativa en la selección del servidor para satisfacer la petición.

Como se aprecia cada una de estas aplicaciones implementan funcionalidades que sirven de guía para desarrollar el Adaptador de Núcleo, cada una de ella la implementan de forma distinta y de acuerdo con las necesidades para las que fueron creadas. Es por ello que al analizarlas se llega a la conclusión de la inexistencia de una herramienta capaz de satisfacer la necesidad del centro, y capaz de implementar por sí sola un componente como el que se propone, que pueda determinar entre varios servidores el más idóneo, es por ello que la implementación de un Adaptador de Núcleo se basaría en cada una de estas funcionalidades detectadas en el análisis realizado sobre estas herramientas mencionadas anteriormente, agrupadas en un solo componente capaz de interactuar con cualquier sistema que solicite su servicio, para su funcionamiento el componente que necesite de su servicio solo tendría que cumplir con los estándares de la arquitectura propuesta, en la que se define la entrada de los datos y las respuestas de salida.

1.4. Conclusiones

En el presente capítulo se concluye que los conceptos asociados al flujo de actividades de los adaptadores de núcleo son de gran importancia debido a que estos permiten profundizar el tema de análisis, con la información brindada permite dar solución a la problemática existente, teniendo en cuenta los aspectos descritos en el capítulo. Por otra parte, el análisis realizado sobre las diferentes tecnologías que implementan funcionalidades que posibilitan partes del flujo de trabajo del componente Adaptador de Núcleo brinda mejor guía en el desarrollo del mismo, permitiendo centrar su desarrollo atendiendo a las deficiencias detectadas con el estudio de dichas tecnologías, proporcionando aspectos necesarios para brindar una solución con buena calidad y mayor número de funcionalidades. Partiendo del estudio realizado sobre los algoritmos de búsquedas heurísticas se determinó que sería factible la utilización del algoritmo A* (A asterisco) debido a las posibilidades que brinda y que están relacionadas con la propuesta de solución de la investigación.

2. CAPÍTULO 2: Lenguajes, metodología y herramientas de desarrollo.

2.1. Introducción.

Los lenguajes, metodologías y herramientas de desarrollo proporcionan los elementos necesarios para sentar las bases en el desarrollo de productos de software eficientes, atendiendo a la selección adecuada de los parámetros que conforman estos tres elementos. Es por ello que en el presente capítulo se realizará una comparación crítica y valorativa sobre estos elementos, así como su correcta fundamentación y que servirá como pilares fundamentales para el desarrollo del componente Adaptador de Núcleo.

2.2. Lenguajes de programación

Un lenguaje de programación no es más que un idioma artificial o una secuencia de símbolos diseñado por el hombre para expresar operaciones que puedan ser llevadas a cabo por una máquina. Puede utilizarse con el fin de crear reglas sintácticas o semánticas, métodos, algoritmos con precisión o como modo de comunicación humana para controlar el comportamiento lógico y físico de una máquina.

C++.

C++ es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C; abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Es un lenguaje multiplataforma, libre e imperativo. En realidad es un súper conjunto de C, que nació para añadirle cualidades y características de las que carecía su predecesor. El resultado es como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para la programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción.

Algunas características fundamentales de C++:

- Es un Lenguaje de programación orientado a objetos.
- Lenguaje muy didáctico, debido a que permite aprender muchos otros lenguajes con gran facilidad, como C#, Java, Visual Basic, Javascript, PHP, entre otros.
- Es muy potente en lo que se refiere a creación de sistemas complejos, un lenguaje muy robusto.
- Actualmente, puede compilar y ejecutar código de C, ya viene con librerías para realizar esta labor.
- Es un lenguaje muy empleado, existen muchos tutoriales en línea, libros y códigos fuentes abiertos.
- Existen muchos algoritmos cuyo pseudocódigo se encuentra ya desarrollado en C++, de manera que puedes tomarlo y amoldarlo a tu solución.

C#

C# nace de Microsoft con la idea de crear un lenguaje mejorado en todos los aspectos. Concebido como lenguaje nativo de su famosa plataforma .Net para aplicaciones web y de escritorio, se ha dicho que C# conjunta principalmente tanto aspectos de C++ como de Java y Visual Basic, pero de una forma más versátil y mejorada agregándole cada vez más elementos que faciliten su uso. C# es un lenguaje muy potente, pero presenta algunas desventajas ya que su principal IDE de programación, Visual Studio es propietario y para su funcionamiento óptimo requiere del framework .Net par Windows y mono para linux.

Java

Java es un lenguaje orientado a objetos, generando ficheros de clases compilados, las cuales son en realidad interpretadas por la máquina virtual de java. Siendo la máquina virtual de java la que mantiene el control sobre las clases que se estén ejecutando.

Para poder trabajar con java es necesario emplear un software que permita desarrollar las aplicaciones. Existen varias alternativas comerciales en el mercado: JBuilder, Visual Age, Visual Café, NetBeans y un conjunto de herramientas shareware, e incluso freeware, que permiten trabajar con java. Pero todas estas herramientas en realidad se basan en el uso de una herramienta proporcionada por Sun, el creador de java, que es el Java Development Kit (JDK). **(Cuernavaca, 2007)**

A pesar de tener muchas ventajas también posee desventajas tales como que dado que la máquina virtual de java es un intérprete y redunda en una falta de rendimiento con relación a aplicaciones equivalentes escritas en código máquina nativo y el poder reducir los problemas de acceso a memoria y liberación

automática hacen de java un lenguaje poco apropiado para desarrollar aplicaciones de base como Sistemas Operativos.

Fundamentación de la selección del lenguaje: C++

Para realizar la implementación del componente Adaptador de Núcleo, se decidió utilizar como lenguaje de programación C++ con el framework QT que utiliza como lenguaje nativo C++, ya que es un lenguaje muy utilizado en aplicaciones cuyo rendimiento es crítico para el funcionamiento óptimo de los sistemas de software, además de las funcionalidades que brinda el lenguaje haciendo uso del framework QT y debido a las ventajas que presenta el framework, que hacen más dinámico el desarrollo del componente mediante el uso de librerías incluidas este. En cuanto a Java las aplicaciones resultantes son más lentas que las desarrolladas con C++, dependen de la instalación de una máquina virtual y en cuánto a C# presenta numerosas ventajas pero tiene el inconveniente de ser un lenguaje propietario atendiendo al proceso de migración actualmente en curso en el país y a las políticas dictadas en el centro.

Framework QT. Principales Características.

QT es un framework para trabajar en C++, es decir, utiliza el lenguaje de programación C++ de forma nativa, otorga muchas ventajas, permite una abstracción del lenguaje C++. Es multiplataforma, permite desarrollar aplicaciones con interfaces gráficas de usuario y programas sin interfaz gráfica como herramientas de la consola y servidores. Posee una API que cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales. Además de ser tan "poderoso", tiene un gran respaldo ya que lleva bastante tiempo en el mercado y posee una gran documentación. Algunas de las aplicaciones que utilizan QT son: KDE, Google Earth, Skype, Qt Extended, VirtualBox, VLC media player, Opera hasta la versión 10.10, entre otras.

Distribuida bajo los términos de GNU Lesser General Public License (y otras), QT es software libre y de código abierto.

Ventajas de QT

- Es multiplataforma.
- Es Software Libre.

- Tiene el apoyo de Nokia.
- Posee cientos de recursos y guías en Forum Nokia.
- Maneja efectos: transparencias, sombras etc, de forma fácil.
- Ide Drag and Drop. Menos código y resultados rápidos.
- Soporte full para la plataforma Symbian.
- Soporta librerías que permiten el acceso a las medias, ficheros, XML, BD, etc.

2.3. Entorno Integrado de Desarrollo (IDE)

Un entorno de desarrollo integrado (IDE) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, puede utilizarse para varios. Los IDE han sido empaquetados como programas de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes.

Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Basic, Object Pascal, etc.

Microsoft Visual Studio

Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones Web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C++, Visual C# y Visual J# utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes aprovechan las funciones de .NET, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y Servicios Web XML.

Este entorno de desarrollo a pesar de ser muy poderoso presenta algunas desventajas como son:

- Es propietario por lo que si vas a desarrollar aplicaciones NET debes comprar VS.NET.
- Requiere de .NET para el desarrollo de aplicaciones.
- No es multiplataforma, solo trabaja en el sistema operativo Windows.

Eclipse

Eclipse presenta código abierto en su entorno de desarrollo, construido principalmente para java. Este fue diseñado originalmente por la empresa IBM y actualmente es desarrollado por la Fundación Eclipse, una comunidad de código abierto.

Su principal desventaja es que consume gran cantidad de recursos en los ordenadores lo que provoca que funcione lento para compilar como para depurar y su instalación es muy tediosa.

QT Creator

Para los desarrolladores, Nokia ofrece Qt Creator, un entorno de desarrollo (IDE) multiplataforma muy completo. Es un IDE creado por Trolltech para el desarrollo de aplicaciones con las bibliotecas Qt, requiriendo su versión 4.x.

Qt Creator es distribuido bajo tres tipos de licencias: Qt Commercial Developer License, Qt GNU LGPL v. 2.1, Qt GNU GPL v. 3.0 y está disponible para las plataformas: Linux, Mac OSX; Windows, Windows CE, Symbian y Maemo.

Qt Creator está totalmente integrado con Qt Designer para ayudarle a diseñar formas de la interfaz de usuario como lo haría con la versión independiente. La integración de Qt Designer incluye también la gestión y finalización del proyecto de código. También viene totalmente integrada con toda la documentación de Qt y ejemplos a través del plugin de Ayuda Qt.

Principales características de Qt Creator:

- Posee un avanzado editor de código C++.
- Posee también una GUI integrada y diseñador de formularios.
- Herramienta para proyectos y administración.
- Ayuda sensible al contexto integrado.
- Depurador visual.
- Resaltado y auto-completado de código.
- Soporte para refactorización de código.

Fundamentación de la selección del entorno de desarrollo: QT Creator

Para el desarrollo del componente Adaptador de Núcleo se decidió utilizar como Entorno de Desarrollo Integrado (IDE) el QT Creator ya que posee una gran documentación y una ayuda avanzada con ejemplos incluidos, es una de las herramientas más utilizadas en los proyectos del centro de GEYSED por lo que

existe una buena capacitación y posee un avanzado editor de código C++. Además, permite la integración de manera sencilla con muchas librerías implementadas en C como la "*libIce",* la cual será utilizada para la comunicación con el componente.

2.4. Metodologías de Desarrollo

Se entiende por metodología de desarrollo a una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software, en Inglés software development methodology (SDM) o system development life cycle (SDLC).

La finalidad de una metodología de desarrollo es garantizar la eficacia (p. ej. cumplir los requisitos iniciales) y la eficiencia (p. ej. minimizar las pérdidas de tiempo) en el proceso de generación de software.

En la actualidad no se debe hablar de una única metodología universal de desarrollo ya que a la hora de crear un proyecto, en dependencia de las características y de la envergadura del mismo es que tiene sentido basarse en la metodología más adecuada para utilizar en ese proceso. El Proceso Unificado Racional (RUP) está basado en UML y es una metodología Orientada a Objetos (OO).

Rational Unified Process (RUP)

RUP es un proceso para el desarrollo de un proyecto de un software que define claramente quien, cómo, cuándo y qué debe hacerse en el proyecto.

La versión de RUP que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo. Dicho proceso tiene tres características fundamentales. La primera de ella es que está dirigido por casos de uso, es decir, que en el proyecto se orientan a la importancia que tiene para el usuario lo que el producto debe hacer. También es un proceso centrado en la arquitectura ya que relaciona la toma de decisiones que indican cómo tiene que ser constituido el sistema y en qué orden se debe hacer. Es iterativo e incremental, divide el proyecto en mini proyectos donde los casos de usos y la arquitectura cumplen sus objetivos de manera más depurada.

RUP se encarga de unificar todo el equipo de desarrollo de software, además de optimizar su comunicación suministrando a cada miembro una aproximación al desarrollo de software con una base de conocimiento de acuerdo con las necesidades específicas del proyecto. Él no es simplemente un proceso,

sino que es un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. Generalmente es aplicado a grandes proyectos de desarrollo de software.

Dentro de sus disciplinas gestiona el control de cambios, que permite mantener al equipo trabajando en los mismos artefactos, en cualquier momento del desarrollo del producto. RUP define como sus principales elementos a los trabajadores, las actividades, los artefactos y los flujos de actividades. Los trabajadores son los propietarios de elementos o artefactos y se encargan de realizar las actividades, las cuales describen cómo una tarea es realizada por un trabajador y a su vez, manipulan elementos. Los artefactos constituyen los productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos. El flujo de actividades describe cuando estas son realizadas por trabajadores y produce un resultado de valor observable.

El ciclo de vida RUP es una implementación del desarrollo en espiral, creado ensamblando los elementos en secuencias semiordenadas, organizando las tareas en fases e iteraciones. Las cuatro fases de RUP son:

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.
- **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene 1 o varios release o versiones del producto que han pasado las pruebas.
- **Transición:** El producto ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

RUP además de estar compuesto por las fases anteriormente mencionadas, también cuenta con 9 flujos de trabajo de los cuales los primeros seis son conocidos como flujos de ingeniería y los tres últimos como de apoyo. En la Figura 5 se muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

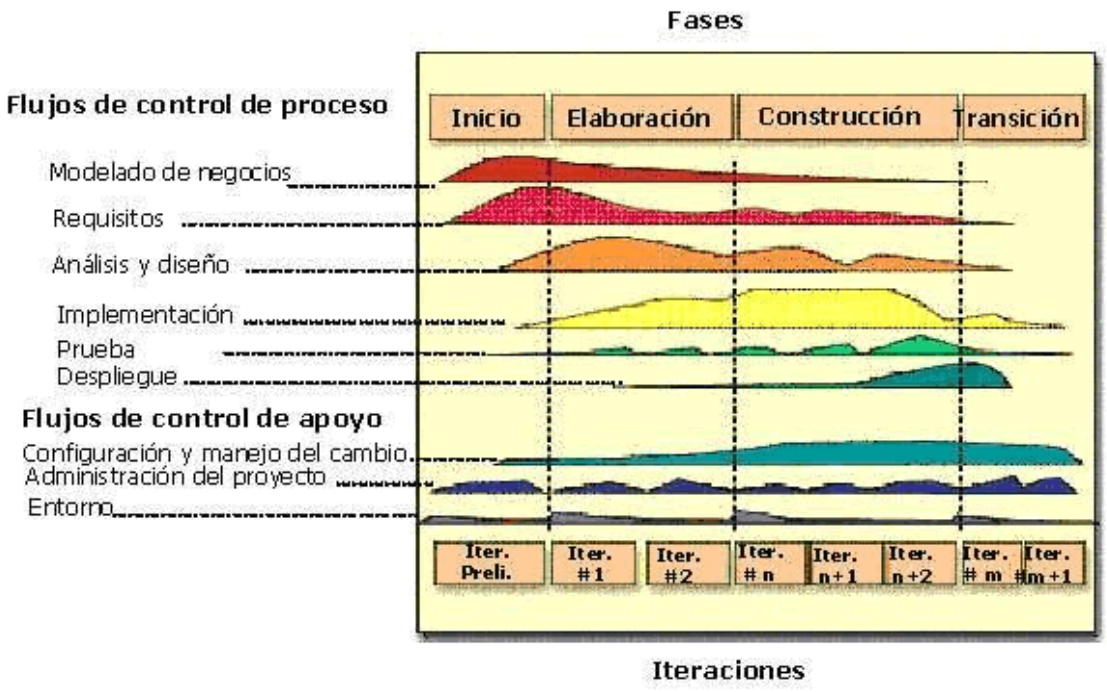

Fig. 5. Fases y Flujos de trabajo según RUP. **(Molpeceres, 2002)**

La metodología RUP es más apropiada para proyectos grandes, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En proyectos pequeños, es posible que no se puedan cubrir los costos de dedicación del equipo de profesionales necesarios.

XP (Xtreme Programming)

XP 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. XP se basa en la 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. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP sin cubrir los detalles técnicos y de implantación de

las prácticas. Posteriormente, otras publicaciones de experiencia se han encargado de dicha tarea. A continuación se presentan las características esenciales de XP organizadas en los tres apartados siguientes: historias de usuario, roles, proceso y prácticas. **(Letelier, y otros, 2006)**

Historias de Usuario.

Son las técnicas utilizadas para especificar los requisitos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. **(Letelier, y otros, 2006)**

Roles XP.

Los roles de acuerdo con la propuesta original de Beck son:

- Programador.
- Cliente.
- Encargado de pruebas (Tester).
- Encargado de seguimiento (Tracker).
- Entrenador (Coach).
- Consultor.
- Gestor (Big boss).

Proceso XP.

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

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.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. **(Letelier, y otros, 2006)**

Concluyendo, en XP, el cliente tiene derechos como por ejemplo decidir qué se implementa, saber el estado real y el progreso del proyecto, añadir, cambiar o quitar requerimientos en cualquier momento,

obtener lo máximo de cada semana de trabajo y obtener un sistema funcionando cada 3 o 4 meses. Además, el desarrollador decide cómo se implementan los procesos, crea el sistema con la mejor calidad posible, pide al cliente en cualquier momento aclaraciones de los requerimientos, estima el esfuerzo para implementar el sistema y cambia los requerimientos en base a nuevos descubrimientos

Lo fundamental en XP es que la comunicación, entre los usuarios y los desarrolladores siempre es íntegra, se logra la simplicidad al desarrollar y codificar los módulos del sistema y la retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

Fundamentación de la selección de la metodología de desarrollo: RUP

RUP cubre todo el ciclo de vida del desarrollo de software, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos, además de otras características que lo destacan. Sin embargo, XP no tiene en cuenta estos aspectos y no le da importancia a la documentación y al lenguaje de modelado. Por otra parte RUP es una metodología que se adapta a los proyectos, permitiendo eliminar los artefactos menos significativos en el desarrollo del sistema, por lo que es la metodología adoptada para documentar todo el proceso ingenieril de la presente investigación. También se hará uso del Lenguaje Unificado de Modelado (UML), el cual es un lenguaje usado para especificar, visualizar y documentar los componentes de un sistema en desarrollo orientado a objetos.

UML

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

Aunque no se considera un estándar oficial es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Este se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software como es el caso de RUP, pero no especifica en sí mismo qué metodología o proceso usar. La sintaxis del lenguaje está compuesta por un conjunto de símbolos con una semántica bien definida para crear diagramas y/o modelos de sistemas. Estos tienen como objetivo fundamental presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

La importancia de UML radica en que posee características visuales que ayudan a los integrantes de un equipo de desarrollo (analistas, diseñadores, especialistas de área y programadores) a intercomunicarse fácilmente, ya sea para establecer colaboración o para facilitar posteriores desarrollos. Se escoge con el objetivo de lograr una mejor estructuración de la documentación presentada y un entendimiento de los procesos desarrollados para personas que puedan integrarse al desarrollo posteriormente, o necesiten comprenderlo para otros fines.

UML presenta las siguientes características:

- Permite modelar sistemas utilizando técnicas orientadas a objetos (OO).
- Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura,
- pruebas, versiones, etc.).
- Cubre las cuestiones relacionadas con el tamaño propio de los sistemas complejos y críticos.
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.
- UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

2.5. Herramientas CASE.

Las Computer Aided Software Engineering o Ingeniería de Software Asistida por Computadora (CASE) son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Se puede ver al CASE como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales.

Las Herramientas CASE fueron desarrolladas para automatizar procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejorados en el ciclo de desarrollo de software.

La mejor razón para la creación de estas herramientas fue el incremento en la velocidad de desarrollo de los sistemas. Por esto, las compañías pudieron desarrollar sistemas sin encarar el problema de tener cambios en las necesidades del negocio, antes de finalizar el proceso de desarrollo.

Rational Rose Enterprise Edition

Es una herramienta que se puede situar dentro del grupo de herramientas más técnicas debido a que se encarga de llevar a cabo tanto la automatización de los sistemas para la posterior generación de código (esto es, realización de los distintos diagramas y generación del código posterior), como para labores de ingeniería inversa, es decir, realización de los diagramas una vez conocido el código. Sin dudas, Rational Rose Enterprise Edition es una forma de ayuda para comprensión del sistema y de sus distintos componentes, y lo mejor es que se puede aplicar ingeniería inversa a una multitud de códigos distintos, siempre que estén orientados a objetos.

La clave está en la creación de componentes, los cuales van a contener una serie de archivos dentro de los cuales se encuentran las distintas clases pertenecientes ha dicho componente. Mediante la especificación de la sintaxis que presentan dichos ficheros, se realiza de forma automática la ingeniería inversa. Se plantea que Rational Rose presenta una pequeña desventaja, y es que necesita de mucha memoria para poder de alguna forma ser manejado de forma rápida y eficiente.

Visual Paradigm

Visual Paradigm es una herramienta CASE que da soporte al modelado visual con UML y con la notación para la gestión de procesos de negocio (BPMN por sus siglas en inglés). Esta permite recorrer el ciclo de vida del desarrollo de software, desde el modelado de negocio hasta el despliegue del producto, generando artefactos en toda su trayectoria. Dentro de las ventajas de esta herramienta se encuentra la rápida construcción de aplicaciones con gran calidad y menor costo. Permite generar diagramas de clases, código desde diagramas y documentación. Proporciona además abundantes tutoriales de UML, así como demostraciones interactivas de dicho lenguaje.

Visual paradigm proporciona soporte a varios lenguajes en generación de código e ingeniería inversa. Lenguajes En generación de Código C++, CORBA IDL, PHP, XML, Schema, Ada y Python.

Presenta algunas características como son:

- Soporte ORM - Generación de objetos Java desde la base de datos.

- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Importación y exportación de ficheros XML.
- Integración con Visio - Dibujo de diagramas UML con plantillas (stencils) de MS Visio.
- Editor de figuras.

Fundamentación de la selección de la herramienta CASE: Visual Paradigm

Visual Paradigm es multiplataforma y gratis, por lo que resulta muy utilizado en el desarrollo de proyectos importantes, además esta herramienta exporta e importa los diagramas con estándar XML y como imágenes, contiene los elementos necesarios para aplicar la metodología RUP ya que es completamente compatible con esta, donde dicha metodología está formada por los elementos necesarios para diseñar e implementar el Adaptador de Núcleo y brinda facilidades para el diseño UML del ciclo de vida de un proceso de desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Además de poseer una amplia bibliografía tanto en materiales audiovisuales como documentación digital. Es una herramienta potente que genera documentación de alta calidad, así como la posibilidad de trabajo con bases de datos. También presenta soporte para la realización de la ingeniería inversa y generación de código.

2.6. Tecnología para la Comunicación.

Conjuntamente con los lenguajes de desarrollo se utilizan tecnologías que apoyadas en las herramientas de desarrollo permiten de manera integral desarrollar el sistema. Para el sistema se necesitan fundamentalmente tecnología de comunicación.

Middleware

El middleware es un software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas.

2.6.1. ICE (Internet Comunication Engine)

En **(Michi Henning, 2009)** se define a ICE como un middleware útil para programadores, libre bajo la licencia GPL, desarrollado por la empresa ZeroC. Es una plataforma de comunicación a través de Internet de alto rendimiento, ligera y abierta, siendo compatible con cualquier tipo de plataforma (Windows, Linux, y otras propietarias), y con los principales lenguajes de programación (C, C++, Java, C#, VisualBasic, Phyton, PHP). La Figura 6 representa un esquema de la arquitectura utiliza en ICE.

ICE emplea el modelo cliente – servidor. Los términos cliente y servidor no están directamente asociados a dos partes distintas de una aplicación, sino que hacen referencia a los roles que las diferentes partes de una aplicación pueden asumir durante una petición **(Michi Henning, 2009)**:

- Los clientes son entidades activas, es decir, emiten solicitudes de servicio a un servidor.
- Los servidores son entidades pasivas, es decir, proporcionan un servicio en respuesta a las solicitudes de los clientes.

Normalmente, los clientes no son clientes puros en el sentido de que solo solicitan peticiones. En su lugar, los clientes suelen ser entidades híbridas que asumen tanto el rol de cliente como el de servidor. De hecho, los sistemas cliente-servidor suelen describirse de una forma eficiente como sistemas peer-to-peer.

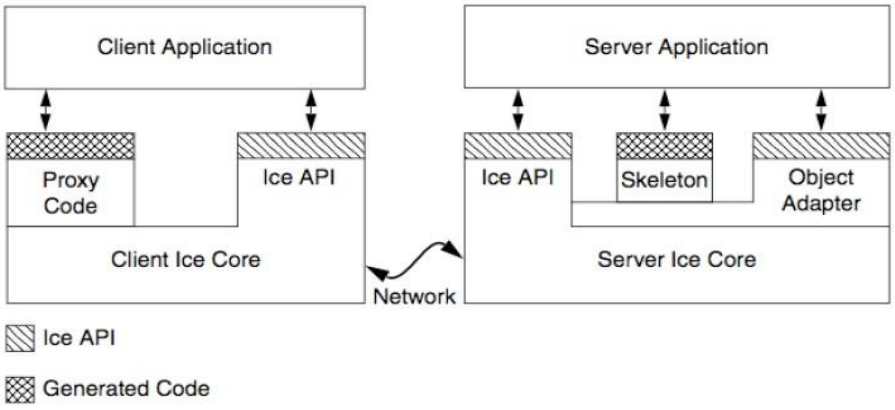

Fig. 6. Arquitectura ICE. (Pérez)

ICE proporciona APIs y librerías para soportar el paradigma cliente/servidor sobre plataformas distribuidas heterogéneas y utilizando múltiples lenguajes. Es orientado a objetos y posibilita que la plataforma sea eficiente en cuanto al uso del CPU, memoria y ancho de banda, incluye una gran variedad de servicios y está formado por los siguientes paquetes:

- **Slice:** El lenguaje de especificación para Ice. Establece el contacto entre servidores y clientes y además es utilizado para describir datos persistentes.
- **Compiladores Slice:** Las especificaciones en Slice se pueden compilar en varios lenguajes de programación: C++, Java, Pitón, PHP, C# y Visual Basic. Los servidores y clientes Ice trabajan juntos, a pesar del lenguaje de programación.
- **Ice:** El núcleo de librerías Ice, entre otras características este paquete gestiona todas las tareas de comunicación utilizando un protocolo altamente eficiente, proporciona soporte de hilos para servidores multihilo y funcionalidades adicionales para soportar escalabilidad extrema con a priori millones de objetos Ice.
- **IceUtil:** Una colección de utilidades como manejo de Unicode y programación con hilos (C++ solo).
- **IceBox:** Un servidor de aplicación específico para aplicaciones Ice. IceBox puede ejecutar y administrar servicios Ice que son cargados dinámicamente como una DLL, librería compartida o clase Java.
- **IceGrid:** Un sofisticado servidor de activación y herramienta de despliegue para computación grid avanzada.
- **Freeze:** proporciona persistencia automática para servants Ice. Con unas pocas líneas de código, una aplicación puede incorporar un vector altamente escalable que gestiona eficientemente objetos persistentes.
- **FreezeScript:** Es necesario para cambiar tipos de datos persistentes, especialmente en proyectos de software grandes. Para minimizar el impacto de estos cambios, FreezeScript proporciona herramientas de inspección y migración para bases de datos Freeze. La herramienta soporta scripts basados en XML ya que es potente y fácil de usar.
- **IceSSL:** Un plug-in de transporte SSL dinámico para el núcleo Ice. Proporciona autenticación, encriptación e integridad de mensaje, utilizando un protocolo SSL estándar.
- **Glaciar:** Algunos retos de los sistemas middleware son la seguridad y los firewalls. Glaciar es una solución firewall de Ice, que simplifica el despliegue de aplicaciones seguras. Autentica y filtra las solicitudes de los clientes y permite realizar retrollamadas (callbacks) al cliente de una manera segura. Utilizado en combinación con IceSSL, proporciona una solución segura que no permite intrusos y es fácil de configurar.

- **IceStorm:** Un servicio de mensajería que soporta la federación. En comparación con otros servicios de mensajería o eventos, IceStorm soporta tipado de eventos, entendiendo que la difusión de un mensaje en una federación es tan fácil como invocar un método sobre un interface.
- **IcePath:** Es un servicio de parcheado para distribuciones software. Mantener el software actualizado es una tarea tediosa, IcePath automatiza las actualizaciones de archivos individuales y jerarquías de directorios. Solo los archivos modificados son descargados en las maquinas clientes, utilizando eficientemente algoritmos de compresión.

Objetos ICE

Un objeto ICE es una entidad conceptual o una abstracción que mantiene una serie de características:

- Un objeto ICE es una entidad en el espacio de direcciones remoto o local que es capaz de responder a las peticiones de los clientes.
- Un único objeto ICE puede instanciarse en un único servidor o, de manera redundante, en múltiples servidores. Si un objeto tiene varias instancias simultáneamente, todavía sigue siendo un único objeto ICE.
- Cada objeto ICE tiene una o más interfaces. Una interfaz es una colección de operaciones soportadas por un objeto. Los clientes emiten sus peticiones invocando dichas operaciones.
- Una operación tiene cero o más parámetros y un valor de retorno. Los parámetros y los valores de retorno tienen un tipo específico. Además, los parámetros tienen una determinada dirección: los parámetros de entrada se inicializan en la parte del cliente y se pasan al servidor, y los parámetros de salida se inicializan en el servidor y se pasan a la parte del servidor. (El valor de retorno es simplemente un parámetro de salida especial).
- Un objeto ICE tiene una interfaz distinguida del resto y conocida como la interfaz principal. Además, un objeto ICE puede proporcionar cero o más interfaces alternativas, conocidas como facetas o facets. De esta forma, un cliente puede seleccionar entre las distintas facetas de un objeto para elegir la interfaz con la que quiere trabajar.
- Cada objeto ICE tiene una identidad de objeto única. La identidad de un objeto es un valor identificativo que distingue a un objeto del resto de objetos. El modelo de objetos definido por ICE

asume que las identidades de los objetos son únicas de forma global, es decir, dos objetos no pueden tener la misma identidad dentro de un dominio de comunicación en ICE.

Proxies

Para que un cliente sea capaz de comunicarse con un objeto ICE ha de tener acceso a un proxy para el objeto ICE. Un proxy es un componente local al espacio de direcciones del cliente, y representa al (posiblemente remoto) objeto ICE para el cliente. Además, actúa como el embajador local de un objeto ICE, de forma que cuando un cliente invoca una operación en el proxy, el núcleo de ejecución de ICE:

1. Localiza al objeto ICE.
2. Activa el servidor del objeto ICE si no está en ejecución.
3. Activa el objeto ICE dentro del servidor.
4. Transmite los parámetros de entrada al objeto ICE.
5. Espera a que la operación se complete.
6. Devuelve los parámetros de salida y el valor de retorno al cliente (o una excepción en caso de error).

Un proxy encapsula toda la información necesaria para que tenga lugar todo este proceso. En particular, un proxy contiene información asociada a diversas cuestiones:

- Información de direccionamiento que permite al núcleo de ejecución de la parte del cliente contactar con el servidor correcto.
- Información asociada a la identidad del objeto que identifica qué objeto particular es el destino de la petición en el servidor.
- Información sobre el identificador de faceta opcional que determina a qué faceta del objeto en concreto se refiere el proxy.

En el desarrollo del componente Adaptador de Núcleo se hace uso de la tecnología de comunicación ICE (Internet Comunication Engine) para el envío de mensajes y la publicación de objetos remotos ya que ICE provee un middleware orientado a objetos para uso en entornos heterogéneos, tiene características que permiten el desarrollo de aplicaciones distribuidas realistas, rehúye de la complejidad innecesaria, provee una implementación eficiente en ancho de banda, uso de memoria y sobrecarga en CPU y posee medidas de seguridad que permiten utilizar las aplicaciones en redes inseguras. ICE garantiza una mayor

estabilidad en las conexiones y una mayor redundancia a los fallos que se puedan ocasionar, resolviendo las caídas de las conexiones de forma íntegra.

2.7. Conclusiones

Para realizar la implementación del componente es necesario realizar la selección de las herramientas, programas y metodologías a utilizar. Como se ha evidenciado a lo largo de este capítulo existen varias herramientas y metodologías de desarrollo, para la correcta selección de las mismas es necesario tener en cuenta los requerimientos del componente con el objetivo de obtener una buena implementación del sistema. Es por ello que se debe realizar un profundo análisis para poder seleccionar estas herramientas y metodologías de desarrollo para el desarrollo del componente AN, por lo que se concluye que para la implementación del mismo se va a utilizar como lenguaje de programación C++ haciendo uso del framework QT, siguiendo como metodología de desarrollo RUP, se define además, la utilización del IDE de desarrollo QT Creator y Visual Parading como herramienta CASE para documentar y modelar los artefactos generados en cada etapa. También se hará uso del middleware ICE para la comunicación entre los componentes con los cuales interactúa el Adaptador de Núcleo. Después de haber hecho una completa selección de las tecnologías y herramientas para desarrollar el presente componente, se está en condiciones de realizar la presentación de la solución propuesta.

3. CAPÍTULO 3: Propuesta y Solución del Componente.

3.1. Introducción

En el presente capítulo se presenta la propuesta de solución con el objetivo de entender el problema de la investigación, el modelo de dominio con la definición de cada una de sus clases, los requerimientos funcionales y no funcionales. Se realiza el diagrama de casos de usos del sistema y la descripción detallada de cada caso de uso. También se detalla la propuesta de selección del algoritmo de búsqueda heurística y se muestran los diagramas de clases del diseño dónde se describe cómo implementar el sistema.

3.2. El modelo de dominio en el desarrollo de software.

El Modelo de Dominio es empleado fundamentalmente cuando los flujos de información son difusos, es decir, que tengan múltiples orígenes y cuando son solo eventos o sucesos. La imposibilidad de realizar subsistemas en el proceso de desarrollo del software dado por las múltiples interconexiones, es uno de los factores que influyen en la decisión de realizar un modelo de este tipo. Además de la existencia del solapamiento o de múltiples responsabilidades y la dificultad en el establecimiento de las reglas del funcionamiento del producto a implementar.

Un Modelo del Dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema, además no incluyen las responsabilidades de las personas que ejecutan las actividades. Según RUP este tipo de modelo representa un subconjunto del Modelo de Negocio, lo cual no significa que siempre que haya un Modelo de Dominio tenga que existir obligatoriamente un Modelo de Negocio. Por lo que el Modelo de Dominio no es más que la representación visual de los conceptos u objetos del mundo real, significativos para un problema o área de interés.

3.2.1. Modelo de dominio

Teniendo en cuenta que no se tienen bien definidos los procesos del negocio se realiza el modelo de clases del dominio y se explica cada uno de los conceptos que forman parte del mismo, además se expone un análisis detallado de los eventos más significativos en el sistema. En la figura 7 se muestra el modelo de dominio correspondiente al componente Adaptador de Núcleo teniendo en cuenta los conceptos y actividades involucradas en ello.

Diagrama de clases del modelo de dominio.

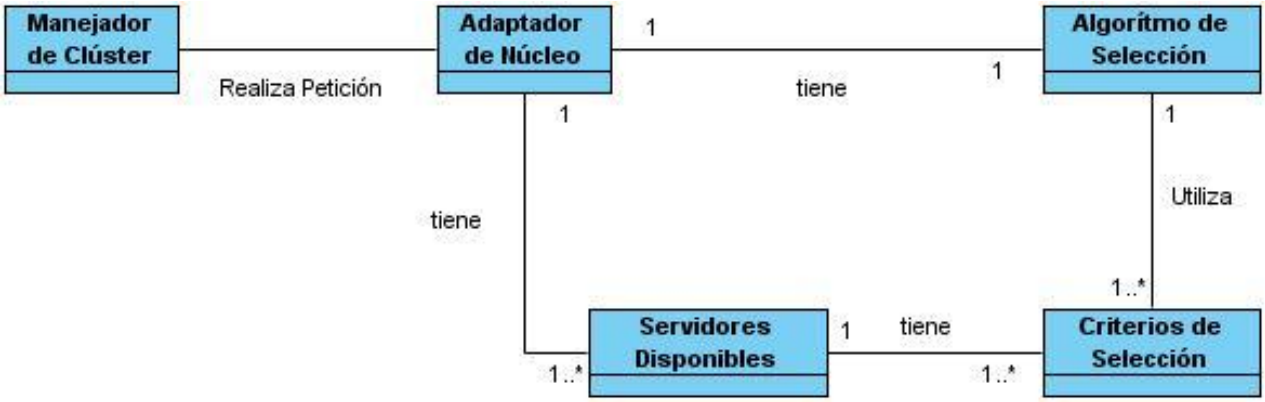

Fig. 7. Diagrama de clases del modelo de dominio

3.2.2. Descripción de las clases del modelo de dominio

A continuación se describen cada una de las clases que interactúan en el dominio del sistema:

Manejador de Clúster (MC).

Descripción de la clase

Sistema que interactúa con el componente Adaptador de Núcleo (AN) el cuál le solicita al componente el servidor idóneo para satisfacer una petición.

Adaptador de Núcleo (AN).

Descripción de la clase

Es el componente que contiene las clases, el algoritmo y los criterios de selección aplicados a los servidores disponibles.

Algoritmo de Selección.

Descripción de la clase

Mediante esta clase se puede obtener el servidor idóneo para satisfacer las peticiones.

Servidores Disponibles.

Descripción de la clase

Contiene toda la información de los servidores donde se van a responder cada una de las peticiones realizadas.

Criterios de Selección.

Descripción de la clase

Son los criterios que contienen los servidores disponibles, éstos se tienen en cuenta para aplicar el algoritmo de selección. Los criterios son: carga del **CPU**[9], memoria **RAM**[10] libre, cantidad de conexiones, cantidad de espacio en disco libre y cantidad de hilos ideales.

3.3. Requerimientos.

Una etapa inicial y muy importante dentro del proceso de la Ingeniería de Software, es la Ingeniería de Requerimientos, donde se lleva a cabo el proceso de descubrir, analizar, escribir y verificar los servicios y restricciones, del sistema de software que se desea producir; este proceso se realiza mediante la obtención, el análisis, la especificación, la validación y la administración de los requerimientos del software. Los requerimientos del software son las necesidades de los clientes, los servicios que los usuarios desean que proporcione el sistema de desarrollo y las restricciones en las que debe operar.

Para la obtención de los requerimientos del componente Adaptador de Núcleo se hizo necesario acudir a varias técnicas relacionadas con dicho proceso como la entrevista a varios líderes del proyecto, la técnica llamada lluvia de ideas, donde cada uno de los integrantes del equipo de desarrollo expresan su punto de vista e ideas relacionadas con las funcionalidades que requiera el producto.

Al realizar el análisis de los requerimientos mediante la técnica de verificación, donde se limaron las ambigüedades y la falta de consistencia, se permitieron llevar a cabo la selección de los requisitos que debe cumplir el componente Adaptador de Núcleo.

3.3.1. Requerimientos funcionales.

Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar, las transformaciones que el sistema realiza sobre las entradas para producir salidas, son capacidades o condiciones que el sistema debe cumplir.

RF 1: Seleccionar servidor idóneo

- El sistema debe permitir seleccionar los servidores idóneos para satisfacer las peticiones realizadas por el Manejador de Clúster (MC). Los servidores idóneos son los servidores con mayor capacidad de procesamiento, esta capacidad de procesamiento es determinada por una heurística

[9] CPU: Central Processing Unit (Unidad de Proceso Central)
[10] RAM: Random Access Memory (Memoria de Acceso Aleatorio)

teniendo en cuenta los criterios de selección (carga CPU, memoria RAM libre, cantidad de conexiones, espacio en disco libre y cantidad de hilos ideales).

RF 2: Consultar servidores disponibles

- El sistema debe permitir consultar los servidores disponibles (clúster) para obtener sus datos y seleccionar el idóneo, y de esta forma dar respuesta a las peticiones realizadas por el MC.

RF 3: Calcular recursos de los servidores

- El sistema debe permitir calcular la cantidad de conexiones, el por ciento de carga del procesador, el porciento de carga de la memoria libre, la cantidad de hilos ideales y el espacio en disco libre que poseen los servidores.

RF 4: Insertar servidores disponibles

- El sistema debe permitir insertar los nodos al listado de servidores disponibles una vez que estos entren en funcionamiento y reporten su estado al Adaptador de Núcleo central.

RF 5: Modificar servidores disponibles

- El sistema debe permitir modificar los parámetros de los servidores disponibles una vez que se le calculen los recursos disponibles para cada servidor.

RF 6: Eliminar servidores disponibles

- El sistema debe permitir eliminar los servidores disponibles una vez que estén fuera de servicio un tiempo determinado.

RF 7: Calcular número saltos

- El sistema debe permitir calcular la distancia (número de saltos) que hay desde los servidores hasta las direcciones ip de las peticiones solicitadas. El número de saltos no es más que la cantidad de subredes por la que tiene que atravesar una petición.

RF 8: Reportar estado

- El sistema debe permitir reportar el estado de los servidores una vez que queden fuera de servicio. Cuando un servidor queda fuera de servicio se le informa al MC el estado del servidor para que el MC migre las peticiones que tiene procesando en este servidor.

3.3.2. Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, se ven como las características que hacen al producto agradable, usable, rápido o confiable.

RNF 1: Fiabilidad

- El sistema debe estar disponible todo el tiempo para los usuarios.

RNF 2: Eficiencia

- Atendiendo a las necesidades del clúster el procesamiento de las peticiones y las respuestas serán atendidas en el menor tiempo posible según las técnicas aplicadas.
- La velocidad de procesamiento de la información, la actualización y la recuperación dependerán de la cantidad de información que tenga que procesar el sistema y la infraestructura tecnológica.

RNF 3: Soporte

- El sistema será escalable permitiendo agregar funcionalidades cuando sea necesario, para así logar mayor extensibilidad y mejores prestaciones.

RNF 4: Restricciones de diseño

- El diseño de la aplicación se hará utilizando la Programación Orientada a Objetos (POO). Encapsulación de la lógica por clases.
- Diseño e implementación de una arquitectura flexible, que permita la fácil integración o desintegración de componentes.
- Se debe emplear el patrón en capas y el patrón orientado a objetos.

RNF 5: Requerimientos de Software

En las PCs Servidor:

- Sistema Operativo: Ubuntu 10.10 o superior.
- Instalación de la librería ICE.

RNF 6: Requerimientos de Hardware

- Se requiere tarjeta de red.
- Memoria RAM de 512 mb o superior.
- Disco duro de 80GB o superior.
- Procesador Intel 3.0 GHz o superior.

3.4. Descripción del sistema. Modelos de Casos de Uso del Sistema.

El Modelo de Casos de Uso del Sistema (MCUS) describe un sistema en cuanto a su utilización, es un modelo que contiene actores, casos de uso y las relaciones que se establecen entre ellos.

Los requerimientos del software se agrupan según sus características, utilidad y usabilidad conformando así los llamados casos de uso, los cuales representan funcionalidades que el sistema ofrece para aportar un resultado de valor para sus actores. Estos casos de uso describen cómo se comporta y funciona un sistema específico.

Los actores se definen como los roles que puede tener un usuario, pueden ser humanos, otros sistemas, máquinas, procesos, hardware, etc. que interactúan con un sistema para de esta forma intercambiar datos. Un actor no es parte del sistema en desarrollo, es un agente externo que interactúa con el mismo en pos de obtener un resultado esperado. El componente Adaptador de Núcleo cuenta con tres actores que se especifican en la tabla 1:

Tabla 1. Descripción del actor del sistema

Actores	Descripción
Manejador de Clúster (MC)	Es el sistema que interactúa con el Adaptador de Núcleo solicitándole Seleccionar un Servidor Idóneo. El sistema le solicita al AN la selección del servidor idóneo para satisfacer una petición.
Reloj	Es el encargado de inicializar los casos de uso Calcular recursos de los servidores, Modificar, Eliminar y Reportar el Estado de los Servidores Disponibles. Cada un intervalo de tiempo determinado Calcula los recursos de los servidores, una vez calculados éstos, se procede a modificar o eliminar servidores disponibles. Si los servidores dejan de funcionar, este se encarga de reportar el estado al manejador de clúster.
Servidor	Es el sistema encargado de Insertar un Servidor Disponible. Una vez que el servidor entra en funcionamiento reporta su estado al Adaptador de Núcleo y lo adiciona a la lista de servidores disponibles.

3.4.1. Modelo de Casos de Usos del Sistema.

Después de haber identificado los actores que interactúan con el sistema y haber sacado el conjunto de funcionalidades escritas en forma de requerimientos que a su vez han sido agrupados en casos de uso según sus peculiaridades, se conforma el Modelo de Casos de Uso del sistema. En la figura 8 se muestra la interrelación del actor del sistema con los casos de uso del sistema.

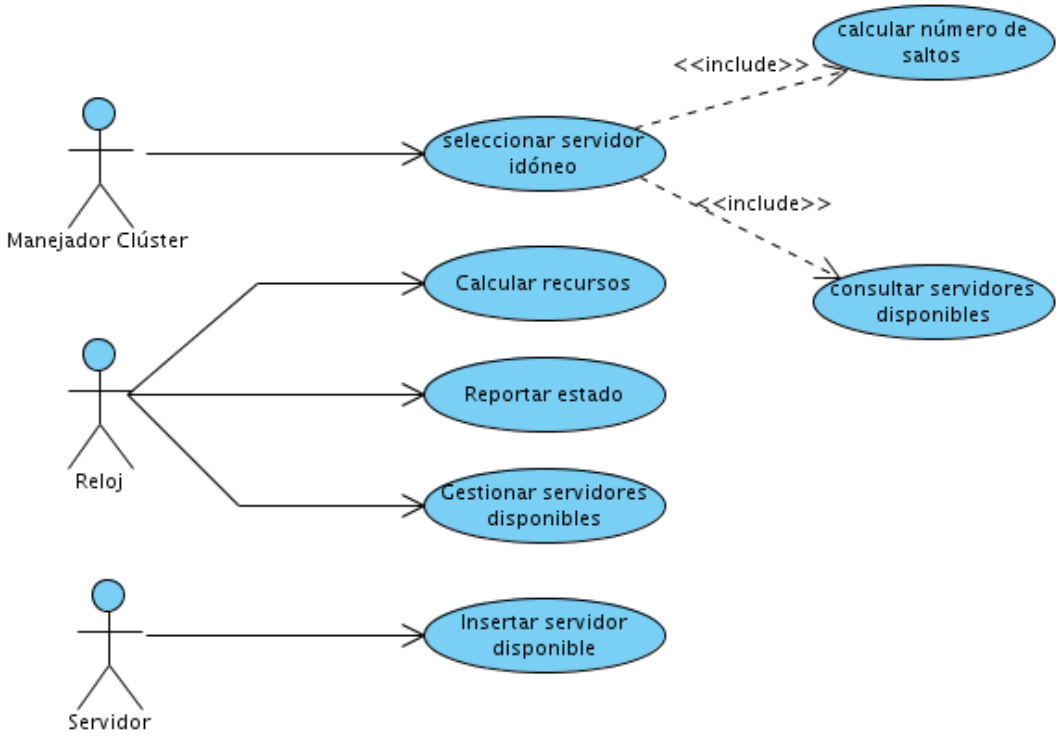

Fig. 8. Diagrama de Casos de Usos del Sistema

3.5. Descripción textual de los Casos de Usos del Sistema.

Para un mayor entendimiento se hace necesario describir textualmente las características particulares de los casos de uso. Además, que para lograr un mayor avance en la construcción del software es recomendable identificar los casos de uso arquitectónicamente significativos, que tienen una prioridad de crítico. A continuación en la tabla 2, se muestra la descripción textual del caso de uso Seleccionar Servidor Idóneo el cual es de vital importancia en el funcionamiento del componente. Las restantes descripciones por su extensión se encuentran en el **Anexo I**

Tabla 2. Descripción del CU Selecciona Servidor Idóneo

Caso de Uso:	Seleccionar Servidor Idóneo.
Actores:	(Manejador de Clúster).
Resumen	El caso de uso se lleva a cabo con el objetivo de seleccionar el servidor idóneo para satisfacer las peticiones realizadas por el manejador de clúster de manera eficiente mediante el empleo de los diferentes criterios de selección.
Precondiciones:	El Adaptador de Núcleo debe publicar el objeto a consumir por el manejador de clúster, quien debe comunicarse con el componente y solicitar un servidor disponible.
Referencias	RF 1
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Manejador de Clúster se comunica con el componente y accede a la interfaz publicada.	1.1. El sistema procesa la petición solicitada por el manejador de clúster. 1.2. Consulta los servidores disponibles y obtiene los criterios de selección. Ver descripción CU Consultar Servidores Disponibles. 1.3. Calcular el Número de Saltos. Ver descripción CU Calcular Saltos. 1.4. Aplica el algoritmo de selección y devuelve el IP del servidor idóneo.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.2. Si no hay servidores disponibles se le notifica al manejador de clúster. 1.3. Muestra un mensaje advirtiendo que no se puede comunicar con los servidores del clúster.

3.6. Propuesta de algoritmo de búsqueda heurística

Al realizar un análisis exhaustivo de los algoritmos de búsquedas heurísticos mencionados en el capítulo 1, se concluye que el algoritmo Escalador de Colina (Hill Climbing) es un algoritmo progresivo, es decir, siempre va hacia adelante, por lo que no tiene vuelta atrás. Al aplicarlo para un número **P** de peticiones selecciona un nodo **N,** al satisfacer la petición **P+1** encontraría una solución si a partir del nodo **N** existieran nuevas soluciones sin volver atrás, por lo que no tiene en cuenta los nodos restantes, siendo así ineficiente para el Adaptador de Núcleo, porque para la petición **P+1** debiera tomar los nodos restantes no seleccionados en la petición **P** como posibles soluciones.

El algoritmo Primero el Mejor (Best-First) al aplicarlo para un número **P** de peticiones selecciona un nodo **N,** para la petición **P+1** tiene en cuenta el nodo **N** y el resto de nodos, pero no tiene en cuenta el costo del esfuerzo para llegar al nodo **N´,** que es el nodo **N** seleccionado para responder la petición **P** con un costo adicional**.** Aplicar este algoritmo no es eficiente ya que el Adaptador de Núcleo debe medir el costo del esfuerzo para llegar a cada **N´** seleccionado.

A* (A asterisco) es un algoritmo completo, siempre que exista una solución dará con ella. Al aplicar este algoritmo para un número **P** de peticiones selecciona un nodo **N,** para la petición **P+1** tiene en cuenta el nodo **N** más el costo del esfuerzo para llegar al nodo **N´**. Es por eso que en la solución propuesta se hace uso de este algoritmo, el cual en su versión básica utiliza una función de evaluación $f(n) = g(n) + h'(n)$, donde g(n) representa el costo asociado a llegar a **N**, adicionándosele así a la función heurística estándar, h'(n), que representa una aproximación de cuan cerca se está del resultado esperado, pero en la presente versión del Adaptador de Núcleo dicha función se presenta como un todo, es decir, g(n) y $h'(n)$ están explícitas en una sola h(n), ya que $h'(n)$ es el esfuerzo de un nodo **N** para llegar **N´** después de haber respondido a una petición **P**, por lo que f(n)=h(n), donde $\boldsymbol{h(n) = \frac{ML+CPDD+HI}{CP+CC+CS}}$ y $h'(n)$ para el nodo **N´** sería **h(N´) – h(N)**, que se encuentra explicita al calcular h(n) para el nodo **N´**.

ML: memoria libre del nodo; CPDD: capacidad de disco duro del nodo; HI: hilos ideales del nodo; CP: carga del procesador del nodo; CC: cantidad de conexiones del nodo; CS: cantidad de saltos desde el nodo a la pc cliente.

3.7. Patrones

Un patrón es un modelo a seguir para realizar una actividad. Estos surgen con la experiencia de los seres humanos de tratar de lograr ciertos objetivos. Por lo tanto, capturan las experiencias existentes y probadas para promover buenas prácticas. Los patrones:

- Son una abstracción de "problema – solución".
- Se ocupan de problemas recurrentes.
- Identifican y especifican abstracciones de niveles más altos que componentes o clases individuales.
- Proporcionan vocabulario y entendimiento común.

3.7.1. Patrones de Arquitectura. Arquitectura en Capas

La arquitectura en capas definida según Garlan y Shaw es una organización jerárquica tal que cada capa proporciona servicios a la capa inmediata superior y se sirve de las prestaciones que le brinda la inmediata inferior. **(David Garlan)**

Las ventajas del estilo en capas son muchas, primero el estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los programadores la participación de un problema complejo en una secuencia de pasos incrementales. En segundo lugar, admite optimizaciones y refinamientos. Proporciona además amplia reutilización, lo que se convierte en uno de los fuertes de esta arquitectura.

En el Adaptador de Núcleo la Arquitectura en Capas se aplica estructurando el trabajo en tres capas, una de ellas representa la interfaz de comunicación, la otra contiene toda la lógica del negocio y por último la tercera contiene todo el acceso a datos.

3.7.2. Patrones de diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Un patrón de diseño es una solución a un problema de diseño. Éstos poseen ciertas características, una de ellas es que son soluciones basadas en la experiencia y que se ha demostrado que funcionan, otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Para realizar el diseño del componente se tuvieron en cuenta varios patrones de diseño, al ser aplicados estos patrones se obtiene como resultado que los diseños sean más flexibles, modulares y reutilizables.

Para asignar las responsabilidades se tuvieron en cuenta algunos de los Patrones GRASP. Según Booch y Rambaugh la responsabilidad es un "contrato u obligación de un tipo de clase". Los patrones de asignación de responsabilidades usados para el desarrollo de la aplicación fueron el patrón Experto, Creador, Alta y Bajo Acoplamiento.

Además de los patrones antes mencionados, también se tuvieron en cuenta algunos de los patrones GOF como son:

Fachada: Tiene como objetivo proporcionar una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La "fachada" satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas.

Este patrón tiene como ventaja que al separar al cliente de los componentes del sistema, se reduce el número de objetos con los que el cliente trata, facilitando así el uso del sistema, además se promueve un acoplamiento débil entre el sistema y sus clientes, reduciéndose las dependencias.

El Adaptador de Núcleo utiliza una clase interfaz publicadora de ICE que encapsula todas las funcionalidades del componente y a través de esta se realiza toda la interacción con los otros componentes.

Singleton: Tiene como objetivo garantizar que una clase tenga una única instancia, proporcionando un punto de acceso global a la misma. **(Larman, 1999)**

Como ventajas se puede señalar que el acceso a la "Instancia_Única" está más controlado y es fácilmente modificable para permitir más de una instancia, y en general, para controlar el número de las mismas (incluso si es variable).

En el Adaptador de Núcleo se aplica en la clase controladora para garantizar que las clases que la instancien accedan al mismo objeto al crear el método estático.

Observador: Define una dependencia "uno-a-muchos" entre objetos, para que, cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente.

Dado que el "observado" no conoce la clase de "observadores" que tiene, el acoplamiento que existe es mínimo y abstracto. Por ello, tanto unos como otros pueden variar y evolucionar en diferentes niveles de detalle de forma independiente.

En el Adaptador de Núcleo se utiliza pues hay clases de la capa del negocio que dependen de la clase de ICE en la capa superior (Presentación) y es necesario que esta notifique en caso de algún cambio.

3.8. Diagramas de Clases del Diseño

El modelo de diseño permite abstraer la implementación del sistema, muestra la relación entre las clases así como los atributos y métodos que las componen y es utilizado como entrada fundamental de las actividades de implementación lo que permite al programador realizar una correcta implementación del sistema. En la figura 9 que se muestran a continuación aparece el diagrama de clases del diseño del caso de uso Seleccionar Servidor Idóneo, los diagramas restantes por su extensión se encuentran en el **Anexo II**. En estos diagramas se muestran los atributos y funcionalidades de las clases, así como la relación que existe entre ellas.

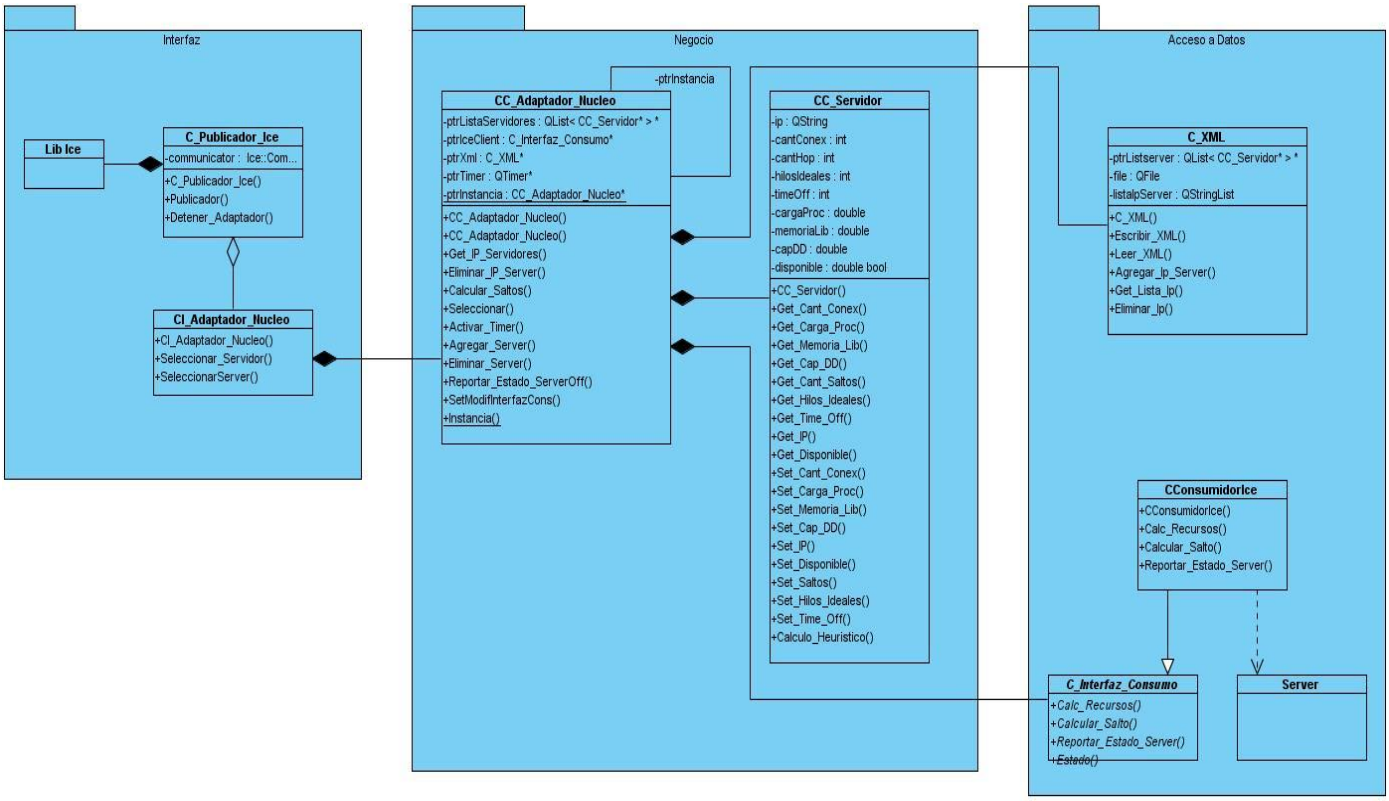

Fig. 9. Diagrama de Clases del Diseño CU Seleccionar Servidor Idóneo

3.9. Conclusiones.

En este capítulo las fases de captura de requisitos y el diseño del Adaptador de Núcleo son muy significativas en el resultado del mismo. Mediante estos flujos de trabajo se puede obtener una óptica de cómo quedará implementado el componente y además permite un fácil entendimiento del sistema. Estas etapas en la vida del software permiten un acercamiento al funcionamiento del sistema. Es por ello que al concluir con estas fases se sientan las bases para dar paso a la implementación del componente, por lo que la captura de requisitos y la realización de las clases del diseño realizadas de forma correcta le brindan un mejor entendimiento al programador.

4. CAPÍTULO 4: Implementación y Prueba.

4.1. Introducción

En el presente capítulo se muestra la vista de implementación donde se desarrollan diferentes artefactos, el diagrama de componentes de código fuente es uno de ellos. Se utiliza para describir la vista de implementación estática de un determinado sistema ya que presenta un nivel más alto de abstracción que un diagrama de clase, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción y eventualmente un componente puede comprender una gran porción de un sistema. Se aprecia además la realización de una serie de actividades (pruebas) encaminadas a encontrar errores en el sistema.

4.2. Modelo de Implementación

En la implementación se parte de los resultados obtenidos en el diseño y se implementa el sistema en términos de componentes, como son los ficheros de código fuente, ficheros de código binario, ejecutables, entre otros. Primeramente, se detalla el modelo de datos del sistema, que es el modelo donde se ve la estructura en la cual se almacenan toda la información requerida en el sistema. Luego se muestra el modelo de implementación que está compuesto por el diagrama de despliegue y el diagrama de componentes. Estos diagramas, describen los componentes a construir, su organización y dependencias entre los nodos físicos en la que funcionará la aplicación.

4.2.1. Diagramas de Componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión de software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. En la figura 15 se muestra el diagrama de componente de código fuente del Adaptador de Núcleo.

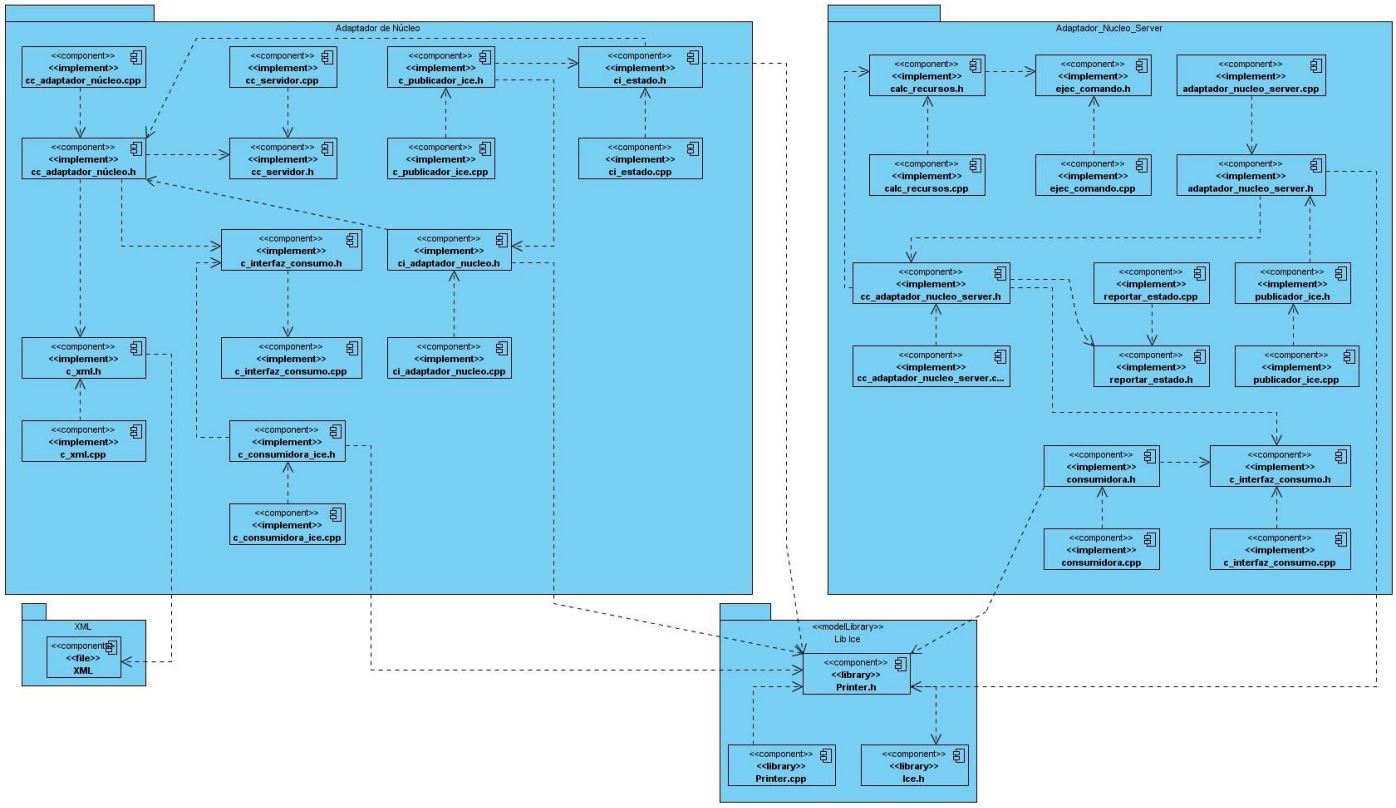

Fig. 15. Diagrama de componente de Código Fuente

4.3. Pruebas

El desarrollo de un producto de software implica la realización de una serie de actividades encaminadas a encontrar errores. Es por ello que se deben incorporar acciones que evalúen la calidad del producto que se está desarrollando. Dentro del proceso de desarrollo de un software, el flujo de trabajo de Prueba, es mediante el que se puede validar que las suposiciones hechas en el diseño y los requerimientos se estén cumpliendo satisfactoriamente, por lo que se encarga de verificar que el producto funcione como se diseñó y que los requerimientos se cumplan adecuadamente.

Este flujo de trabajo brinda soporte para encontrar, documentar y solucionar defectos en el sistema, por lo debe estar presente en todo el ciclo de vida del desarrollo del sistema para ir refinándolo paulatinamente y no al final del mismo. Prueba es un proceso de ejecución de un programa con la intención de descubrir un error no detectado hasta entonces. Los casos de pruebas no pueden asegurar la ausencia de errores, solo puede demostrar que existen defectos en el software.

Las pruebas se pueden realizar basándose en dos esquemas diferentes: demostrar a través de pruebas de caja blanca que las operaciones internas se ajustan a lo especificado y que los componentes internos andan bien, o mediante las pruebas de caja negra, conociendo la función del programa, intentar demostrar que las funciones están correctas.

4.3.1. Prueba de caja blanca

Las pruebas de caja blanca están dirigidas principalmente a las funciones internas del sistema. Se realizan probando los caminos lógicos del sistema y examinando el estado del programa en varios puntos para determinar si el estado real coincide con el esperado. Para realizar estas pruebas es necesario conocer la lógica del programa.

Mediante los métodos de prueba de la caja blanca, el ingeniero de software puede obtener casos de prueba que garanticen que:

1. Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
2. Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsa.
3. Ejecuten todos los bucles en sus límites y con sus límites operacionales.
4. Se ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. **(Pressman, 2000)**

Prueba del camino básico.

La prueba del camino básico es una técnica de prueba de la Caja Blanca propuesta por Tom McCabe. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.

4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. En la Fig. 16 se muestra una porción del código programado al que se le realizarán las pruebas de caja blanca. Este código corresponde al caso de uso Seleccionar_Servidor, el mismo es el responsable de seleccionar un servidor para satisfacer una petición realizada por el cliente al manejador de clúster.

```
QString CC_Adaptador_Nucleo::Seleccionar(QString pIpCliente)
{
    QString mejorSolucion = "No hay servidores disponibles";
    qreal solucionParcial = 0, solucion = 0;                          1

    if (ptrListaServidores->length()==1 && ptrListaServidores->at(0)->Get_Disponible()) ---->2
    {
        mejorSolucion=Get_IP_Servidores().at(0); ---->3
    }
    else ----> 4
    {
        for(int j=0; j < ptrListaServidores->length(); j++) ----> 5
            {
                CC_Servidor * server = ptrListaServidores->at(j); ----> 6
                if (server->Get_Disponible()) ----> 7
                {
                    int saltos=ptrIceClient->Calcular_Salto(pIpCliente,server->Get_IP())

                    qreal huristica=server->Calculo_Heuristico();                          8

                    solucionParcial = huristica/(saltos+1);

                    if (solucion < solucionParcial) ----> 9
                    {
                        solucion = solucionParcial;
                        mejorSolucion = server->Get_IP();              10
                    } ----> 11
                } ---->12
            } ----> 13
    } ----> 14
    return mejorSolucion; ----> 15
}
```

Fig. 16. Representación del código del método Seleccionar_Servidor.

La siguiente figura muestra el grafo de flujo representado para el código anterior:

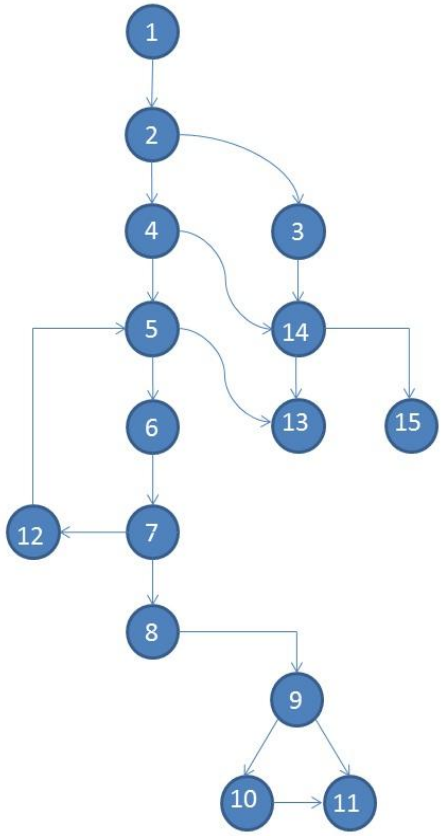

Fig. 17. Grafo del flujo asociado al método Seleccionar_Servidor.

Cálculo de la complejidad ciclomática:

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y proporciona un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente.

Complejidad ciclomática [V (G)] = Cantidad de Aristas [A] – Cantidad de nodos [N] + 2.

V (G) = A – N + 2

V (G) = 18 – 15 + 2

V (G) = 5

El cálculo realizado arrojó como resultado 5, por lo que se puede plantear que la complejidad ciclomática del código anteriormente expuesto es de 5, lo que significa que existen cinco posibles caminos por donde el flujo puede circular, representando este valor el límite mínimo del número total de casos de pruebas que se deben realizar para el procedimiento tratado. En la tabla 9 se muestran los caminos seleccionados para realizar los casos de pruebas.

Tabla 3. Caminos básicos del flujo.

Número	Camino básico
1	1-2-3-15
2	1-2-4-5-6-7-8-9-10-11-12-5-13-14-15
3	1-2-4-5-6-7-12-5-13-14-15
4	1-2-4-5-6-7-8-9-11-12-5-13-14-15
5	1-2-4-5-13-14-15

Luego de tener elaborado el Grafo de Flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Para realizar los casos de pruebas es necesario tener en cuenta los siguientes aspectos.

- **Descripción**: Se realiza la entrada de datos necesaria para validar que los parámetros obligatorios no pasen nulos al método.
- **Entrada**: Se exponen los parámetros que son necesarios para realizar el método.
- **Resultados Esperados**: Se define el resultado esperado que debe devolver el método.

En la tabla 10 se muestra el caso de prueba para el camino 1-2-3-15, los casos de pruebas para los restantes caminos se encuentran en el **Anexo III**.

Tabla 4. Caso de prueba para el camino básico 1.

Caso de prueba para el camino básico 1	
Camino 1	1-2-3-15
Descripción	Los datos de entrada cumplirán con los siguientes requisitos: La lista de servidores disponibles solo contiene un solo servidor y está disponible, los métodos que se llaman toman valores y es pasado por parámetro el IP cliente.
Entrada	pIpCliente=10.34.13.222 cantidad de Servidores=1 ipServer=10.34.13.220 disponible=true mejor solución=10.34.13.220
Resultados esperados	Se devuelve como mejor solución el ip del único servidor que existe en el listado de servidores disponibles.

Después de haberse aplicado las pruebas, en este caso pruebas de caja blanca, en específico las pruebas del camino básico en las que se selecciona diferentes caminos a través del cálculo de la complejidad ciclomática a partir de un segmento de código, se arriba a la conclusión de que los resultados obtenidos fueron aceptables, pues se pudo comprobar que el flujo de trabajo es correcto para las funciones probadas y que cumple con las condiciones planteadas.

4.4. Conclusiones

En el presente capítulo se presenta una visión global de cómo quedó finalmente implementado el sistema, expresado en componentes de implementación, lo que permite tener una óptica de la estructura del código fuente del mismo. Se obtuvo como resultado que todos los componentes se implementaron cumpliendo con esta etapa de desarrollo. Además, fueron realizadas las pruebas al componente, las que demostraron que el componente cumple con las funcionalidades y condiciones necesarias planteadas anteriormente.

CONCLUSIONES GENERALES

Con el desarrollo del presente trabajo de diploma, se cumplió con las tareas y objetivos trazados para desarrollar el componente Adaptador de Núcleo. El mismo cumple todos los requisitos planteados para su desarrollo por lo que se convierte en un componente cuyas funcionalidades básicas muestran los resultados esperados.

Se generó la documentación relacionada con el desarrollo del componente, la cual permitirá comprender y reutilizar el sistema desarrollado. También se implementó un componente capaz de realizar la selección del nodo para darle respuesta a determinadas peticiones, logrando que el sistema esté preparado para evolucionar hacia nuevas funcionalidades y actualizar las existentes.

El componente fue desarrollado mediante aplicaciones libres por lo que puede ser desplegado sobre entornos libres, permitiéndole a Cuba contar con una herramienta propia que responda a los intereses de sus organizaciones y permita desarrollarlas.

Es una aplicación novedosa y no se tienen referencias de otras con características similares. Su importancia radica en que independiza el trabajo de los servidores de streaming distribuidos dándoles la posibilidad de aprovechar al máximo los recursos disponibles en los servidores donde son desplegados sus componentes.

RECOMENDACIONES

Luego de haber concluido el sistema propuesto y cumplido los objetivos tratados se plantean las siguientes recomendaciones:

- Realizar la implementación del Adaptador de Núcleo haciendo uso de IceGrid.
- Aplicar otros criterios de selección para seleccionar el servidor idóneo.
- Tener en cuenta el tipo de petición para seleccionar el servidor idóneo.
- Realizarle al componente otros tipos de pruebas.

GLOSARIO DE TÉRMINOS

Multimedia: Se aplica a cualquier objeto que utilice simultáneamente diferentes formas de contenido informativo como texto, sonido, imágenes y video para informar o entretener al usuario.

TDT: Televisión Digital Terrestre.

SSD: Servidor de Streaming Distribuido.

UCI: Universidad de las Ciencias Informáticas.

TIC: Tecnologías de la Información y las Comunicaciones.

GEYSED: Centro Geoinformática y Señales Digitales.

Servidor: Software u ordenador que provee servicios a otros programas o equipos denominados clientes.

Cliente: Es una aplicación informática cuya función es acceder a los servicios que ofrece un servidor, haciendo uso generalmente de una red de telecomunicaciones.

Multiplataforma: Se utiliza el término para denominar a los programas, lenguajes de programación u otra clase de software que pueden brindar sus prestaciones funcionando sobre diversas combinaciones de hardware y software.

Aplicación: Es una clase de programa informático creado para facilitar al usuario un determinado tipo de trabajo. Esto lo caracteriza frente a otros programas como los sistemas operativos, las utilidades y los lenguajes de programación.

Protocolo: En informática es un método estándar que permite la comunicación entre procesos. Comprende un conjunto de reglas y procedimientos que deben respetarse para el envío y la recepción de datos a través de una red.

XML: Lenguaje extensible de Marcado, o en sus siglas en inglés: Extensible Markup Language.

UML: Lenguaje Unificado de Modelado.

GRASP: Patrones de Software para la asignación General de Responsabilidad.

GOF: Gang-of-Four “Pandilla de los cuatro”.

RUP: Proceso Unificado de Rational, o en sus siglas en inglés: Rational Unified Process.

API: Application Programming Interface - Interfaz de Programación de Aplicaciones es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Frameworks: Estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

CPU: Central Processing Unit (unidad de proceso central). La CPU es el cerebro del ordenador, es donde se producen la mayoría de los cálculos. En términos de potencia del ordenador, la CPU es el elemento más importante de un sistema informático.

RAM: Random Access Memory (Memoria de acceso aleatorio). Un tipo de memoria de ordenador a la que se puede acceder aleatoriamente; es decir, se puede acceder a cualquier byte de memoria sin acceder a los bytes precedentes.

Clúster: Grupo de múltiples ordenadores unidos mediante una red de alta velocidad, de tal forma que el conjunto es visto como un único ordenador, más potente que los comunes de escritorio.

Licencia GNU/GPL: Está diseñada para asegurar la libertad de distribuir copias de Software Libre (y cobrar por ese servicio), asegurar que recibirá el código fuente del programa o bien podrá conseguirlo si quiere, asegurar que puede modificar el programa o modificar algunas de sus piezas para un nuevo programa y para garantizar que puede hacer todas estas cosas.

TRABAJOS CITADOS

Alvarez, Miguel Angel. 2001. DesarrolloWeb.com. [En línea] 2001. [Citado el: 04 de 12 de 2010.] http://www.desarrolloweb.com/articulos/482.php.

Bijit, Leopoldo Silva. *Algoritmos heurísticos.* s.l. : UNIVERSIDAD TECNICA FEDERICO SANTA MARIA.

Borges, Jorge Luis. PARADIGMA HEURISTICO. [En línea] [Citado el: 15 de 01 de 2011.] http://wilucha.com.ar/Paradigma/A_ParaHeuris.html.

Cuernavaca, Morelos. 2007. Curso de Java. [En línea] 2007. [Citado el: 3 de 12 de 2010.] http://tikal.cifn.unam.mx/~jsegura/LCGII/java3.htm.

David Garlan, Mary Shaw. *"An introduction to software architecture". CMU Software Engineering Institute Technical Report.* CMU/SEI-94-TR-21.

Dayrel Almaguer Chávez, Dysán García Riera. 2008. *Cluster de servidores de bases de datos para aplicaciones web, sobre software libre.* 2008.

Durán, M.R. *Sistema Automatizado para la Empresa Cubana TELECABLE .*

Fajardo, Yaily González. 2007. *ESTUDIO DE LAS TECNOLOGIAS DE VIDEO Y SERVIDORES DE STREAMING EN.* 2007.

Fernández, Raúl Castro. *Desarrollo de software de videovigilancia para sistemas embarcados distribuidos con ICE.*

Larman, Craig. *UML y Patrones.*

—. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Mexico : s.n., 1999.

Letelier, Patricio y Penadés, Mª Carmen. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : s.n.

Letelier, Patricio y Penadés, MaríaCarmen. 2006. Métodologías ágiles para el desarrollo de software:. [En línea] 2006. [Citado el: 7 de 12 de 2010.] http://barranquillo.ucaldas.edu.co/rgarcia/ingsoft/recursos/dorada/masyxp.pdf.

Marcos Guglielmetti, Analia Lanzillotta. 2005. Mastermagazine. *Definición de Streaming.* [En línea] 2005. [Citado el: 27 de Noviembre de 2010.] http://www.mastermagazine.info/termino/6781.php.

María F. Rosique Contreras, Pedro Sánchez Palma. 2003. *La programación distribuida de aplicaciones desde la perspectiva de la programación orientada a aspectos.* Cartagena : s.n., 2003.

Michi Henning, Mark Spruiell. 2009. *Distributed Programming with Ice.* 2009.

Pérez, José Carlos Cortizo. *Programación Concurrente y Distribuida. CORBA.* Madrid : s.n.

Pressman. 2000. *Ingeniería de Software. Un enfoque práctico.* 2000.

2010. Tecnología de la Información y la Comunicación. [En línea] 26 de 11 de 2010. http://web.educastur.princast.es/proyectos/grupotecne/asp1/tic/vermensajebbb.asp?idmensaje=4753.

Torres, Angel Laguna. 2009. *Propuesta de Servidor Streaming de software libre para la captura y transmisión de video y sonido digital.* 2009.

Tusch, Roland, y otros. *An Adaptive Distributed Multimedia Streaming Server in Internet Settings.* Australia : s.n.

Tusch, Roland, y otros. 2004. *Offensive and Defensive Adaptation in Distributed.* 2004.

VILMA, QUISPE CARITA, HARRY, HUAMANTUCO SOLORZANO DANTE y LUIS, VARGAS YUPANQUI JOSE. 2011. *MONOGRAFIA METODOLOGIA RUP (RATIONAL UNIFIED PROCESS).* Peru : s.n., 2011.

BIBLIOGRAFÍA CONSULTADA

[En línea] [Citado el: 03 de 12 de 2010.] http://www.alegsa.com.ar/Dic/lenguaje%20de%20programacion.php.

[En línea] [Citado el: 15 de 01 de 2011.] http://www.zeroc.com/doc/Ice-3.3.1/manual/IceGrid.40.4.html#76700.

Apple Computer Inc. About Darwin Streaming Server en. [En línea] http://www.publicsource.apple.com/projects/streaming/.

CASE, Herramientas. [En línea] [Citado el: 7 de 12 de 2010.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rea_c_ji/capitulo2.pdf..

Comunidad de Programadores. [En línea] [Citado el: 3 de 12 de 2010.] http://www.lawebdelprogramador.com/news/mostrar_new.php?id=227&texto=C+sharp&n1=134702&n2=2&n3=0&n4=0&n5=0&n6=0&n7=0&n8=0&n9=0&n0=0.

Danae. [En línea] [Citado el: 25 de Noviembre de 2010.] danae.rd.francetelecom.com/technology-Distributed-Adaptation.htm.

Eva. [En línea] [Citado el: 25 de 1 de 2011.] http://eva.uci.cu/file.php/102/Curso_2010-2011/Clases/Semana_06/Conf_7/Materiales_complementarios/Introduccion_a_la_Disciplina_de_Requisitos.pdf.

Fajardo, Yaily González. 2007. *ESTUDIO DE LAS TECNOLOGIAS DE VIDEO Y SERVIDORES DE STREAMING EN.* 2007.

Fernández, Raúl Castro. *Desarrollo de software de videovigilancia para sistemas embarcados distribuidos con ICE.*

Flumution. *Flumotion Streaming Server.* [En línea] [Citado el: 25 de Noviembre de 2010.] http://www.flumotion.com/esp/server_tecnologia.php?menu=2.

IGNetwork. [En línea] [Citado el: 6 de 12 de 2010.] http://foro.ignetwork.net/showthread.php?15188-IDE-Entorno-integrado-de-desarrollo-(Concepto-importante).

Larman, Craig. *UML y Patrones.*

—. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Mexico : s.n., 1999.

Letelier, Patricio y Penadés, Mª Carmen. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : s.n.

Letelier, Patricio y Penadés, MaríaCarmen. 2006. Métodologías ágiles para el desarrollo de software:. [En línea] 2006. [Citado el: 7 de 12 de 2010.] http://barranquillo.ucaldas.edu.co/rgarcia/ingsoft/recursos/dorada/masyxp.pdf.

Loja, Universidad Católica. [En línea] [Citado el: 15 de 1 de 2011.] http://www.utpl.edu.ec/eva/descargas/material/175/G18401.8.pdf.

Marcos Guglielmetti, Analia Lanzillotta. 2005. Mastermagazine. *Definición de Streaming.* [En línea] 2005. [Citado el: 27 de Noviembre de 2010.] http://www.mastermagazine.info/termino/6781.php.

María F. Rosique Contreras, Pedro Sánchez Palma. 2003. *La programación distribuida de aplicaciones desde la perspectiva de la programación orientada a aspectos.* Cartagena : s.n., 2003.

Michi Henning, Mark Spruiell. 2009. *Distributed Programming with Ice.* 2009.

MSDN. [En línea] [Citado el: 6 de 12 de 2010.] http://msdn.microsoft.com/es-es/library/fx6bk1f4(v=vs.80).aspx.

Paraiso Linux. [En línea] [Citado el: 6 de 12 de 2010.] http://paraisolinux.com/herramientas-para-modelado-uml/.

Pérez, José Carlos Cortizo. *Programación Concurrente y Distribuida. CORBA.* Madrid : s.n.

Pressman. 2000. *Ingeniería de Software. Un enfoque práctico.* 2000.

Principiantes. *Servidor de Radio Digital.* [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.principiantes.info/util/crear_radio_internet.php.

Programación en Castellanos. [En línea] [Citado el: 6 de 12 de 2010.] http://www.programacion.com/noticia/qt_creator-_un_completo_entorno_de_desarrollo_1723.

Rational. [En línea] [Citado el: 6 de 12 de 2010.] http://www.insight.com.uy/consultoria-rational-que-es-rational.html.

Sitio Oficial de Subsonic. [En línea] [Citado el: 24 de Noviembre de 2010.] http://www.subsonic.org/.

Sitio Oficial de VLC. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.videolan.org/vlc/.

2010. Tecnología de la Información y la Comunicación. [En línea] 26 de 11 de 2010. http://web.educastur.princast.es/proyectos/grupotecne/asp1/tic/vermensajebbb.asp?idmensaje=4753.

Tusch, Roland, y otros. *An Adaptive Distributed Multimedia Streaming Server in Internet Settings.* Australia : s.n.

Tusch, Roland, y otros. 2004. *Offensive and Defensive Adaptation in Distributed.* 2004.

Utilización de programación funcional distribuida y clusters Linux en el desarrollo de servidores de vídeo bajo demanda. [En línea] [Citado el: 25 de Noviembre de 2010.] http://www.madsgroup.org/vodka/papers/sit01.pdf.

VideoStreaming. [En línea] [Citado el: 27 de Noviembre de 2010.] http://www.upm.es/sfs/Rectorado/Vicerrectorado%20de%20Tecnologias%20de%20la%20Informacion%20y%20Servicios%20en%20Red/Gabinete%20de%20Tele-Educacion/Perfil%20PDI/Videostreaming.pdf.

Visual Paradigm. [En línea] [Citado el: 5 de 12 de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

Yahoo respuestas. [En línea] [Citado el: 3 de 12 de 2010.] http://es.answers.yahoo.com/question/index?qid=20080818183756AAG6VRL.