# UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

## Facultad de Ciencias y Tecnologías Computacionales


**Título:** Componente de navegación sobre los flujos de video transmitidos en vivo por las cámaras IP.

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

**Autor:** Yunior Luis Rosabal Peña

**Tutores:** MSc. Reinier Pupo Ruiz

Ing. Reidel Morales Toledo

**La Habana, 2017**

**"Año 59 de la Revolución"**

“Debemos procurar una revolución en la tecnología que nos dé invenciones y maquinarias que inviertan las tendencias destructivas que ahora nos amenazan a todos.”

Ernst Friedrich Schumacher

# *Declaración de Autoría*

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste se firma la presente a los ____ días del mes de _____ del año _____.

_____

**Autor:** Yunior Luis Rosabal Peña

_____ _____

**Tutor:** Ing. Reidel Morales Toledo **Tutor:** MSc. Reinier Pupo Ruiz

**Tutor: MSc Reinier Pupo Ruiz**

Máster en Informática Aplicada. Alcanzó el Nivel Superior en el año 2009 y el título de máster en el 2015 en la Universidad de las Ciencias Informáticas, La Habana, Cuba. Cuenta con 8 años de experiencia en el desarrollo de aplicaciones para el procesamiento de videos y sistemas de video vigilancia dentro del Centro Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas. Se desempeña actualmente como desarrollador en el sistema de video vigilancia Suria.

Correo electrónico: rpupo@uci.cu

**Tutor: Ing. Reidel Morales Toledo**

Ingeniero en Ciencias Informáticas, alcanzó el Nivel Superior en el año 2015 en la Universidad de las Ciencias Informáticas, La Habana, Cuba. Cuenta con 2 años de experiencia en el desarrollo de aplicaciones para el procesamiento de videos y sistemas de video vigilancia dentro del Centro Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas. Se desempeña actualmente como desarrollador en el sistema de video vigilancia Suria.

Correo electrónico: rlcordero@uci.cu

# *Dedicatoria*

*Durante estos cinco años en la universidad he vivido experiencias únicas, unas muy buenas, otras no tanto. Hoy quiero agradecer aquellas personas que me han apoyado y ayudado en mi formación.*

*Primeramente, a mis padres por toda la confianza y el apoyo que me dado.*

*A mis hermanas por estar siempre cuando las necesito*

*A mis tías por sus consejos que me han ayudado mucho, sobre todo en mi carrea.*

*A George que siempre ha sido como un tío para mí.*

*A mis sobrinos y mis primos que siempre me motivan a seguir a delante.*

# *Agradecimientos*

*Primeramente, le agradezco a mi familia por todo su apoyo incondicional.*

*A mis amigos por toda la ayuda que me han brindado.*

*A mi tutor y a Rolando por dedicarme un poco de su tiempo y brindarme su apoyo.*

*A mi oponente y al tribunal que gracias a sus consejos pude seguir a delante con la tesis.*

## Resumen

Suria es un sistema de video vigilancia soportado sobre tecnología IP (*Internet Protocol*), el cual permite la gestión de un conjunto de cámaras de seguridad de diferentes modelos, así como acciones sobre las mismas en dependencia de las capacidades de cada una. A este sistema es necesario incorporarle un mecanismo que permita la navegación sobre los flujos de video transmitidos en vivo por las cámaras IP. Para ello se desarrolló un componente con el empleo del *framework* multimedia GStreamer. El uso de este componente permite regresar la transmisión en tiempo real y observar lo que pasó instantes antes para analizar con más detalle el video. Además, contribuye a evitar la pérdida de información ante situaciones en la que se haya desatendido la pantalla. El estudio de las principales soluciones a nivel nacional e internacional no permitió encontrar un componente similar que pudiera integrarse con el sistema de video vigilancia Suria. El componente fue desarrollado utilizando la metodología AUP-UCI (*Agile Unified Process)* y el lenguaje de programación C++. Para comprobar el buen funcionamiento del componente de navegación se realizaron pruebas de estrés y rendimiento, las cuales permiten ver el comportamiento del mismo en los momentos de carga extrema, y permite probar cuán rápida y eficiente puede ser la respuesta de este en un entorno controlado donde se utilizará.

**Palabras clave:** cámara IP, navegación, tiempo real, video.

## Abstract

Suria is a surveillance system supported on IP (Internet Protocol) technology, which allows the management of a set of security cameras of different models, as well as actions on them depending on the capabilities of each. To this system it is necessary to incorporate a mechanism that allows the navigation on the flows of video transmitted live by the IP cameras. For this, a component was developed using the multimedia *framework* GStreamer. The use of this component allows to return the transmission in real time and observe what happened moments before to analyze the video in more detail. In addition, it helps prevent the loss of information in situations where the screen has been unattended. The study of the main solutions at national and international level did not allow finding a similar component that could be integrated with the Suria video surveillance system. The component was developed using the AUP-UCI methodology (Agile Unified Process) and the C ++ programming language. In order to verify the good functioning of the navigation component, tests of stress and performance were carried out, which allow to see the behavior of the same in the moments of extreme load, and allows to test how fast and efficient the response of this in a controlled environment where It will be used.

**Keywords:** camera IP, navigation, time real, video.

# Índice General

## Índice de Figuras

# Índice de Tablas

## Introducción

La video vigilancia IP[1] es una tecnología de vigilancia visual donde se combinan los beneficios analógicos de los tradicionales CCTV (Circuitos Cerrados de Televisión) con las ventajas digitales de las redes de comunicación IP, permitiendo la supervisión local o remota de imágenes y audio. Inicialmente estos sistemas fueron concebidos para la detección de intrusos en espacios cerrados o abiertos, permitiendo también agilizar el control ante hurtos u otros incidentes. El uso combinado de la video vigilancia con una fuerza policial de reacción rápida, amplía el alcance de la policía y permite un mejor servicio a un costo reducido. (Kumar, y otros, 2015)

Los primeros sistemas de video vigilancia se montaban sobre tecnología analógica, podían estar compuestos por una o más cámaras de vigilancia conectadas a uno o más monitores de video, que reproducían las imágenes capturadas por las cámaras. Para tener un control sobre estos sistemas los usuarios debían monitorear los medios a proteger durante largos períodos de tiempo. Para el almacenamiento de las grabaciones estos sistemas utilizaban video casetes, por lo que cuando hacía falta realizar alguna revisión del material, tenían que observarse las grabaciones por completo. Este método hacia muy compleja la búsqueda de hechos específicos y muy poco manipulables las cámaras. (Kruegle, 2007)

Actualmente los sistemas de vigilancia modernos poseen avances destacados en comparación con los de antaño, tales como la inclusión de cámaras IP, capacidades inalámbricas que eliminan el tendido de cables e inclusión de sistemas de inteligencia para el tratamiento del video que posibilitan controlar la gestión de eventos. Además el uso de cámaras IP permite al usuario del sistema acceder al video en tiempo real en cualquier momento y desde cualquier ordenador (Meissner, 2005). Para visualizar la interfaz de la cámara, el usuario solo debe desde un explorador web escribir la dirección IP de la misma y de esta manera accede a su sistema de navegación.

El sistema de navegación de cada cámara varía en dependencia del modelo de la misma, algunas incluyen más funcionalidades que otras, tales como sensores, enfoque, inclinación y zoom. De manera general cuando reproducimos un video podemos navegar sobre el mismo a través de su barra de progreso. Esta no es más que una línea de tiempo que orienta al usuario sobre el progreso del archivo.

[1] ***Internet Protocol*** **o Protocolo de Internet (IP):** Es un protocolo de comunicación de datos digitales clasificado funcionalmente en la Capa de Red según el modelo internacional OSI.

Esta línea de tiempo puede ser utilizada para avanzar o retroceder dentro del video, simplemente haciendo clic en la posición donde el usuario quiera situar la reproducción o arrastrando el botón de la misma. (Solano, 2012)

La gran mayoría de los reproductores de video y música del mundo permiten la navegación sobre sus contenidos, permitiendo ir hacia adelante y hacia atrás durante la reproducción. Para esta funcionalidad el reproductor cuenta con una barra de progreso que contiene la duración del video, y así hacer más fácil la búsqueda de un contenido específico en el archivo que se está reproduciendo. Los sistemas de video vigilancia, por lo general, permiten el análisis de las grabaciones realizadas por las cámaras de seguridad, permitiendo la navegación dentro de sus grabaciones para detectar la ocurrencia de eventos significativos. (MEPSD, 2008)

Suria es un sistema de video vigilancia soportado sobre tecnología IP, desarrollado en la Universidad de las Ciencias Informáticas (UCI) en el Departamento de Componentes del Centro de Geoinformática y Señales Digitales (GEySED). Esta solución de software permite la grabación, visualización, recuperación, análisis y gestión de los flujos de videos provenientes de las cámaras IP. Este sistema está formado por 8 módulos: Interfaz general, Grabación, Análisis, Servidor de Administración, Administración[2], Recuperador y Visor.

El módulo Visor se encarga de visualizar los flujos de videos capturados por las cámaras, así como realizar acciones sobre las mismas en la medida de las capacidades de cada una (mover, hacer zoom, grabar, tomar instantánea, patrullar). Este módulo ofrece varias vistas que varían según la cantidad de cámaras activas, permitiendo la visualización simultánea de hasta 16 cámaras. Además, este módulo es el que estará en constante interacción con el usuario del sistema. Actualmente, cuando se está visualizando una cámara solo se puede ver lo que está ocurriendo en ese preciso instante, por lo que si es necesario observar lo que pasó minutos antes para analizar con más detalle el video u ocurre alguna situación en la que se haya desatendido la pantalla no se puede regresar atrás en la transmisión, lo que provoca pérdida de información en un sistema cuyo principal objetivo es informar de cualquier incidente que pueda afectar la seguridad. Además, el no percatarse de eventos significativos que hayan acontecido previamente, ofrece una idea equivocada de lo que puede estar pasando en la realidad y por tanto se dificulta la toma de decisiones.

[2]**Administración:** Incluirá como parte de él el módulo Autonomía

La situación descrita anteriormente lleva a plantearse el siguiente **problema a resolver:** ¿Cómo visualizar los eventos ocurridos previamente durante las transmisiones en vivo de las cámaras IP en el sistema video vigilancia Suria? Para realizar un estudio de la problemática y desarrollar la investigación se define como **objeto de estudio:** La navegación durante la transmisión en vivo. Enmarcándose como **campo de acción:** La navegación durante la transmisión en vivo de flujos de video provenientes de las cámaras IP en el sistema video vigilancia Suria. Para dar solución a la situación problemática descrita y al problema planteado se define como **objetivo general:** Desarrollar un componente de navegación sobre los flujos de video transmitidos en vivo por las cámaras IP, que permita navegar sobre una transmisión en vivo del sistema video vigilancia Suria.

Para guiar la investigación se definen las siguientes **preguntas científicas:**

- ✓ ¿Qué es la navegación durante una transmisión en vivo?
- ✓ ¿Qué tecnologías se deben utilizar para permitir el desplazamiento en el tiempo en una transmisión en vivo?
- ✓ ¿Qué procesos intervienen en la navegación sobre los flujos de video transmitidos en vivo por las cámaras IP?

Con el fin de dar cumplimiento al objetivo general, se definen las siguientes **tareas de investigación:**

1. Caracterizar los procesos relacionados a la navegación en una transmisión en vivo.
2. Caracterizar las herramientas y tecnologías a utilizar en el desarrollo del componente de navegación sobre los flujos de video transmitidos en vivo por las cámaras IP.
3. Realizar el análisis y diseño de los artefactos que se involucran en la construcción del componente.
4. Implementar la herramienta que posibilite la navegación sobre los flujos de video transmitidos en vivo por las cámaras IP.
5. Validar a través de las pruebas de software el correcto funcionamiento del componente.

### Métodos de investigación:

### Métodos Teóricos

- ✓ ***Histórico/lógico:*** Este método permite realizar un estudio acerca de las posibles soluciones que permiten la navegación en los videos de las cámaras en tiempo real en un sistema de video

vigilancia; sus antecedentes y tendencias actuales a lo largo de la historia, así como los conceptos y metodologías de desarrollo existentes para llevar a cabo su construcción. (Gomez Bastar, 2012)

- ✓ ***Analítico-Sintético:*** Este método se emplea para realizar un estudio de la bibliografía referente a los diferentes conceptos asociados al tema, permitiendo la extracción de los elementos más importantes que se relacionan con la navegación en tiempo real sobre un flujo de video. (Gomez Bastar, 2012)

### Métodos empíricos

- ✓ ***Entrevista:*** Es la técnica de obtención de información a través del diálogo, por lo que requiere de ciertas habilidades por parte del entrevistador. La entrevista utilizada es no estructurada debido a que es aplicada para conocer el objeto de investigación desde un punto de vista externo, sin que se requiera aún la profundización en la esencia del fenómeno, por lo que las preguntas a formular por el entrevistador, se deja a su criterio y experiencia (Gomez Bastar, 2012). Además, se utiliza con el objetivo de definir las principales funcionalidades que debe cubrir el componente a implementar y determinar la importancia del desarrollo de este componente, así como los beneficios que aportaría al proyecto.

La presente investigación está distribuida en los siguientes capítulos:

**Capítulo 1: Fundamentación teórica.** En este capítulo se fundamentan términos de importancia para la investigación. Se realiza un estudio del arte sobre sistemas que existen en Cuba y el mundo con características similares a las requeridas para la investigación. Además, se analiza el objeto de estudio y se seleccionan las tecnologías, el lenguaje de programación, las herramientas y la metodología para el desarrollo de la solución.

**Capítulo 2: Análisis y diseño del sistema.** En este capítulo se describe el modelo del dominio del sistema que se va a desarrollar. También se identifican los requisitos funcionales y no funcionales. Se definen los casos de usos y actores del sistema. Se especifica la arquitectura base de la aplicación, mostrándose los artefactos generados en esta fase, así como los patrones de diseño utilizados.

**Capítulo 3: Implementación y prueba.** En este capítulo se aborda el desarrollo de la solución, mostrándose los artefactos generados por la metodología utilizada. Se muestra el diagrama de despliegue y el de componentes de implementación. También se realizan las descripciones de las pruebas y se exponen los resultados de las mismas.

# Capítulo 1: Fundamentación Teórica

## Introducción

Este capítulo tiene como objetivo realizar una introducción al contenido de la investigación, haciendo referencia a conceptos asociados al dominio del problema a resolver. Se realiza un estudio sobre soluciones similares existentes en Cuba y el mundo. Además, se mencionan las principales herramientas y tecnologías que serán utilizadas en el desarrollo del componente de navegación.

## 1.1 Términos asociados al dominio de la investigación

### *Video vigilancia*

Se entiende por video vigilancia la utilización de imágenes de video provenientes de las cámaras, ya sea en tiempo real o en visualización de grabaciones, para funciones de vigilancia de incidentes de seguridad. Las cámaras pueden encontrarse interconectadas mediante cableado o inalámbricamente a una estación central de control la cual es la encargada de administrar el sistema y almacenar los datos. (ESYS, 2016)

Un sistema de video vigilancia IP permite supervisar video y grabarlo desde cualquier lugar de la red, tanto si se trata por ejemplo de una red de área local(LAN) o de una red de área extensa(WAN) como Internet. Esto permite una monitorización remota en tiempo real, centralizando las labores de monitorización, almacenamiento y gestión en un central de alarmas ubicada en un emplazamiento diferente al del espacio monitorizado. (García Mata, 2011)

Suria, por su parte es un sistema de video vigilancia profesional, el cual permite la gestión de cámaras de seguridad independientemente del modelo que sea. Además, incorpora una amplia gama de video sensores que pueden aplicarse tanto a las transmisiones en vivo como a las grabaciones, así como una gran variedad de reglas de decisión que dotan al sistema de un alto grado de autonomía.

### *Cámara IP*

La cámara IP es un dispositivo que se utiliza principalmente para redes de video vigilancia, permiten realizar una emisión configurada según unas determinadas características y ser consultadas desde miles de kilómetros de distancia a través de Internet. Estas cámaras se integran y gestionan como un dispositivo más de una red de telecomunicaciones. Disponen de software propio, con aplicaciones específicas para manejarlas. Estas aplicaciones ofrecen al usuario la posibilidad de consultar, en cualquier momento, las

imágenes capturadas por las cámaras a través de Internet mediante su dirección IP. (Gallego, y otros, 2011)

La presente investigación centra su enfoque en este tipo de cámaras debido a que Suria es un sistema de video vigilancia que opera sobre redes digitales, por lo tanto, los flujos de video digital procesados en el sistema provienen de cámaras IP.

***Video***

Un video es un sistema de grabación y reproducción de imágenes, que pueden estar acompañadas de sonidos. Consiste en la captura de una serie de fotografías (en este contexto llamadas "fotogramas") que luego se muestran en secuencia y a gran velocidad para reconstruir la escena original. En un sistema de video vigilancia IP, representan la señal de video emitida por las cámaras y es el resultado final que es mostrado en las estaciones de monitorización, grabado para su posterior análisis o procesado en busca de información de interés para el usuario. (CCM Benchmark group, 2016)

***Transmisión en vivo (live streaming)***

El streaming[3] en directo o en vivo es aquel que transmite eventos que están sucediendo justo en el momento de la difusión. Por ejemplo, la transmisión de conciertos o clases, son eventos que típicamente se difunden usando este tipo de streaming. La transmisión de radio y televisión por Internet también tienen estas características, aunque en ocasiones, la información que se difunde no parte de un evento en directo (Novoa, 2007). El streaming en vivo es utilizado por las cámaras IP para enviar sus flujos de video a través de este.

***Navegación en video***

La navegación en el video se realiza a través de una barra de progreso. Esta le permite al usuario ser capaz de controlar la reproducción, moviéndose hacia adelante y hacia atrás en el video. La barra de progreso ubicada en el área de control de reproducción proporciona una visualización numérica del rango de tiempo actual del video, y una serie de controles de reproducción; permitiéndole al usuario desplazarse con mayor precisión en el video a la hora de buscar un momento específico dentro de la grabación. (Rosenfeld, y otros, 2003)

[3] **Streaming o Emisión:** técnica que permite reproducir audio o video, sin necesidad de descargarla previamente, ya se trate de acontecimientos que suceden en ese momento (o con un desfase mínimo) o de material pregrabado.

## 1.2 Descripción del objeto de estudio

Para la realización de esta investigación es necesario profundizar en los procesos de captura y almacenamiento de flujos de video, los cuales intervienen en el proceso de navegación en tiempo real sobre los mismos.

### *1.2.1 Captura de flujos de video provenientes de cámaras IP*

La captura de un flujo de video es el proceso mediante el cual se obtiene el video proveniente de un dispositivo que se encuentre transmitiendo, ya sea una transmisión en tiempo real o bajo demanda. (Schulzrinne, y otros, 1998)

**Protocolo RTSP**

El protocolo RTSP establece y controla tanto uno como varios *streams* sincronizados de datos multimedia, como pueden ser el audio y el video. No efectúa el envío de los datos, aunque el envío de información de control en medio de la transmisión de datos es posible. Lo que hace el RTSP es el control remoto a través de la red de los servicios de datos multimedia. En este protocolo no hay conexiones, lo que se tiene son sesiones mantenidas por el servidor. Una sesión RSTP no está ligada a una conexión de nivel de transporte. Esto significa que, durante una sesión RTSP, se pueden abrir y cerrar tantas sesiones de transporte como sea necesario. También se puede utilizar UDP como protocolo de transporte (Ordinas, 2008). Algunos modelos de cámaras IP incorporan este protocolo para transmitir los flujos de video capturados.

**Captura de un flujo de video RTSP**

El proceso de obtención de un flujo de video vía RTSP se basa en peticiones cliente-servidor. A continuación se describe este proceso (Schulzrinne, y otros, 1998):

1. El cliente accede a la URL[4] RTSP utilizando el nombre del servidor, el puerto y el nombre de acceso al flujo de video específico que desea capturar. También la URL RTSP puede contener, en caso de ser necesario, usuario y contraseña para acceder al flujo.
2. El cliente realiza una conexión con el servidor, utilizando el protocolo TCP.

[4] ***Uniform Resource Locator*** **(URL):** Localizador de Recursos Uniforme. Es un identificador formado por una secuencia de caracteres, de acuerdo a un formato y estándar, que designa recursos en una red (Masinter, y otros, 2010).

3. Cuando la conexión está establecida correctamente, el cliente envía al servidor una petición OPTIONS con datos necesarios para obtener el flujo.
4. El servidor devuelve información que incluye la fecha, el número de sesión, el nombre del servidor y los métodos soportados.
5. El cliente envía una petición DESCRIBE para obtener una descripción del flujo. El servidor responde con todos los valores necesarios para la inicialización.
6. El cliente envía una petición SETUP para el flujo de datos que se quiere reproducir. El SETUP especifica los protocolos aceptados para el transporte de los datos del video.
7. El cliente inicializa los programas requeridos para reproducir el flujo o procesarlo.
8. El cliente envía una petición PLAY para que el servidor comience a transmitir el video.
9. Cuando el cliente ha terminado con el flujo de video, envía un SET_PARAMETER que contiene las estadísticas de la sesión, seguido de una petición TEARDOWN para dar por terminada la conexión con el servidor.

*1.2.2 Almacenamiento de flujos de video provenientes de cámaras*

El almacenamiento de un flujo de video no es más que crear en el disco duro de la computadora un fichero que contenga los datos de los flujos de video provenientes de las cámaras. Este fichero puede tener diferentes formatos dependiendo de la calidad y el tamaño con que se desea guardar el video. En los sistemas de video vigilancia, se almacenan los flujos de video emitidos por las cámaras para su posterior análisis. (Schulzrinne, y otros, 1998)

*1.2.3 Navegación sobre los flujos de video transmitidos por las cámaras IP*

La navegación sobre un video no es más que desplazarse hacia el instante de tiempo que se desee a través de una barra de progreso que permite moverse sobre el mismo además de indicar el tamaño del archivo y el instante de tiempo en que se encuentra la reproducción. Este proceso durante una transmisión en vivo conlleva la captura y almacenamiento de los flujos de video recibidos para poder después moverse sobre esos videos.

Esto se debe a lo explicado anteriormente, que en el caso de las transmisiones en vivo la información se muestra sin tener la necesidad de descargarla y si los videos no están grabados físicamente no es posible desplazarse en su contenido y saber lo que ha acontecido previamente. En el caso del sistema de video

vigilancia Suria, es necesaria la navegación sobre los flujos de video transmitidos en vivo por las cámaras IP permitiendo el desplazamiento sobre los búferes de video mostrados con anterioridad por las cámaras. Este proceso incluirá la captura, almacenamiento y reproducción de los flujos de video emitidos por los dispositivos de seguridad.

### 1.3 Características del sistema de video vigilancia Suria

El sistema de video vigilancia Suria está soportado sobre tecnología IP y cuenta hoy con dos versiones, una para sistemas operativos basados en GNU/Linux y otra para el sistema operativo Windows. Como objetivo general el proyecto se propone desarrollar una solución de software que permita la grabación, visualización, recuperación, análisis y gestión de los flujos de videos provenientes de cámaras IP. Este software posee la capacidad de visualizar, analizar, recuperar, administrar y grabar los flujos de videos generados por las cámaras de seguridad, para lograr, de tal modo, que se requiera menos esfuerzo para garantizar la protección de las instituciones. El sistema se estructura básicamente en 8 módulos:

**Servidor de Administración:** Es el módulo fundamental del sistema. Todo el tráfico de información pasa por él, facilitando el control y supervisión, permitiendo además la flexibilidad de la aplicación. Es el único que interactúa directamente con la base de datos.

**Administración:** Es el módulo encargado de gestionar los usuarios, cámaras, zonas, roles y permisos en el sistema, así como la planificación de tareas de grabación y análisis. Contiene las funcionalidades del subsistema de Autonomía, controlando la capacidad en disco en el almacén de grabaciones, capaz de tomar decisiones de eliminar, mover, o copiar archivos, según las reglas definidas por el usuario.

**Análisis:** Es el módulo encargado de realizar el procesamiento a los flujos de videos que le sean asignados ya sea por planificación o bajo demanda.

**Grabación:** Es el módulo encargado de realizar las grabaciones ya sea por planificación o bajo demanda.

**Interfaz General:** Es el módulo que permite el control de acceso al sistema. Además, según el rol y los permisos del usuario, le brinda la funcionalidad de acceder a los distintos módulos de la aplicación y establecer la comunicación con el servidor.

**Recuperador:** Es el módulo encargado de la recuperación y la visualización de videos grabados. Permite

además realizar búsquedas de videos almacenados en el servidor, ya sea por zona, cámara, fecha, evento, entre otras. Igualmente proporciona la facilidad a los usuarios que tengan acceso a sus funcionalidades de realizar análisis de los videos grabados.

**Visor:** Es el módulo que permite visualizar los flujos de videos capturados por las cámaras. Pueden existir varias instancias de esta estación corriendo en los dominios físicos del sistema. Tiene la capacidad de reflejar todo el aspecto organizativo con que se manejan las cámaras internamente, además de poder visualizarlas de manera independiente o colectiva (a manera de vistas). También permite manipularlas en la medida de las capacidades de cada una.

## 1.4 Estado actual de la temática

Luego de un estudio bibliográfico se caracterizaron varios sistemas que trabajan con video en vivo y de una manera u otra utilizan la navegación sobre videos, centrando el estudio en sistemas de video vigilancia. A continuación, se describen las principales características de varios de estos sistemas.

### *1.4.1 Soluciones que permiten la navegación en vivo en Cuba*

#### ***DATYS***

En Cuba, la empresa Desarrollo de Aplicaciones, Tecnologías y Sistemas (DATYS), dedicada al desarrollo de aplicaciones informáticas, cuenta entre sus productos con *Xyma Safe Vision*, software de video vigilancia profesional basado en tecnología IP que tiene como objetivo el monitoreo en tiempo real y la vigilancia de instalaciones y exteriores. Este software permite realizar análisis sobre las grabaciones, permitiendo la navegación sobre las mismas para facilitar el proceso. Sin embargo, esta acción solo se puede realizar sobre videos grabados, no sobre las transmisiones en vivo. (DATYS, 2016)

En el estudio de esta solución no fue identificada ninguna funcionalidad que pueda servir de base en el desarrollo de la presente investigación. El software *Xyma Safe Vision* no cuenta con un sistema que se encargue de la navegación en tiempo real de los flujos de video provenientes de las cámaras IP y por tanto no representa una solución viable para Suria.

#### ***Cajas decodificadoras de televisión digital***

Es un dispositivo diseñado para captar las señales digitales y convertirlas al formato analógico. Además, permiten al usuario no solo recibir con mayor calidad la señal radio-televisiva sino que incorpora nuevas prestaciones como: PVR[5] y *TimeShift*. El PVR permite la grabación a demanda y posterior reproducción en la propia caja de la programación televisiva. La función *TimeShift*, que significa desplazamiento en el tiempo, es una de las novedades de la televisión digital, que permite, una vez activada, hacer pausa, ir hacia atrás y hacia adelante, las veces que desee en una transmisión de televisión en vivo. Para activar esta función debe tener conectado en el puerto USB un dispositivo de almacenamiento masivo, memoria flash o disco externo con capacidad, en el cual se creará un fichero temporal. El usuario tiene la opción de salvar este fichero en memoria (Soto Villalobos, 2017). Esta solución no se puede utilizar en el sistema de video vigilancia *Suria* debido a que es un dispositivo electrónico y no se cuenta con el código fuente del mismo, provocando que este no se pueda integrar. Aunque este producto no pueda ser integrado, permitió identificar el uso del *framework* multimedia *GStreamer* y el lenguaje de programación C++ que serán utilizados para el desarrollo de un componente que permita la navegación sobre un flujo de video en vivo.

### *1.4.2 Soluciones que permiten la navegación en vivo en el mundo*

#### *iSpy*

iSpy es un software de código abierto válido tanto para usuarios domésticos que únicamente necesiten una o pocas cámaras de vigilancia como para negocios que necesiten un sistema de vigilancia a pequeña escala para ser controlado localmente. Se trata de un programa informático totalmente gratuito que se puede descargar sin coste alguno desde su página web. Una de sus principales características es que permiten la navegación sobre las grabaciones, permitiendo desplazarse en el tiempo, pero solo sobre las grabaciones, además de permitir la monitorización de tantas cámaras como se desee. También, permite grabar video y audio a demanda, incluso programando horarios (Velasco, 2014). Este software no permite la navegación en tiempo real, pero si reafirma el uso del *framework GStreamer* para la transmisión y procesamiento de los videos.

#### *Ksenos*

Ksenos es un software de software de gestión de video fácil de usar hecha en Finlandia. Es adecuado para sistemas de video vigilancia de todos los tamaños. Los sistemas pueden consistir en uno o más

[5] ***PVR***: *Personal Video Recorder* o Grabador de Video Personal

servidores y clientes remotos opcionales. Tanto las cámaras IP como analógicas pueden conectarse a los servidores Ksenos. Este software de video vigilancia cuenta con una innovadora línea de tiempo que se encuentra siempre visible y muestra grabaciones o imagen en vivo. Esta línea de tiempo tiene una funcionalidad que permite al usuario navegar hacia una posición puntual que desee dentro del video. Además, Ksenos permite la exportación de videoclips a soportes de grabación externos. Se pueden incluir varias cámaras de diferentes períodos de tiempo en un clip de video (Ksenos, 2016). El software Ksenos, aunque permita la navegación sobre los flujos de video no es factible utilizarlo en el sistema Suria porque permite realizar esta navegación solo si la cámara está grabando y no sobre los flujos de video transmitidos en tiempo real.

***Vivotek***

VIVOTEK es el software profesional de gestión de video diseñado para gestionar todos los productos de vigilancia IP de VIVOTEK con funciones intuitivas y numerosas. Este sistema de video vigilancia cuenta con la funcionalidad de grabación con cambio de tiempo (*Time-shift*). Es una característica extremadamente útil que permite que el sistema de VIVOTEK solicite y grabe video almacenado en caché en dispositivos (cámaras y servidores de vídeo) sólo cuando ocurre un evento. Por lo tanto, puede ahorrar ancho de banda de red cuando no se producen eventos. Después de que la memoria caché de flujo esté habilitada en un dispositivo, almacenará video / audio en la memoria incorporada del dispositivo mientras haya capacidad disponible (cuando la memoria esté llena, el video grabado más antiguo será borrado) (Howard, 2010). Esta solución no es viable para el sistema de video vigilancia *Suria* debido a que no se cuenta con el código fuente de la misma, provocando que esta no se pueda integrar. Además, aunque este sistema no pueda ser integrado, permitió identificar uno de los lenguajes de programación utilizado para el procesamiento sobre un flujo de video en vivo, el cual es de utilidad para el desarrollo de componente.

### *1.4.3 Conclusiones sobre las soluciones existentes*

Luego de una investigación realizada a nivel nacional e internacional se puede concluir que, en Cuba a pesar de existir empresas dedicadas al desarrollo de sistemas de video vigilancia, no se encontró una solución que permita la navegación en tiempo real de los flujos de video. La mayoría de los productos investigados a nivel internacional son softwares privativos por lo que no se puede obtener su código fuente, lo que imposibilita su integración con *Suria*, además el uso de estos requiere un costo adicional al

desplegar el sistema. Otros productos, como el caso de la caja decodificadora, no cuenta con una interfaz de integración que permita incorporarla al sistema de video vigilancia Suria. A pesar de que los productos investigados, no sirven como solución a la problemática planteada, el estudio de sus funcionalidades permitió identificar el uso del *framework* multimedia *GStreamer* para la navegación sobre el flujo de video en tiempo real.

Por lo anteriormente planteado, se concluye que las soluciones analizadas no satisfacen las necesidades de Suria. Por tanto, se requiere crear un componente de navegación para las transmisiones en vivo de los flujos de video provenientes de las cámaras IP, en el sistema de video vigilancia Suria.

## 1.5 Herramientas y tecnologías para el desarrollo del software

Teniendo en cuenta la necesidad de un componente de navegación para el sistema de video vigilancia Suria, se empleará para su desarrollo las herramientas y las tecnologías que se definen en este proyecto, con el objetivo de lograr una fácil integración con sus módulos.

### *1.5.1 Metodologías de Desarrollo*

La metodología de desarrollo se identifica como el conjunto de procedimientos, técnicas y soporte documental utilizados para el diseño de sistemas de información. En ingeniería de software cuando se hace referencia al desarrollo de software, se está hablando del desarrollo de programas, los cuales deben cumplir una serie de etapas o fases, para poder funcionar con otros métodos ya establecidos en otras disciplinas de ingeniería (Sommerville, 2010).

Para el desarrollo de la presente investigación se escoge como metodología de desarrollo AUP-UCI, ya que es la definida por el proyecto para el cual se desarrolla el componente de navegación, además genera los artefactos que permiten obtener la documentación necesaria para una mejor comprensión de la herramienta a desarrollar.

**Proceso Unificado Ágil (AUP-UCI)**

El Proceso Unificado Ágil de Scott Ambler o *Agile Unified Process* (AUP) en inglés, es una versión simplificada del Proceso Unificado de Racional (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Al no existir una metodología de software universal, ya que toda

metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo) exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, la cual se llama Ejecución y se agrega la fase de Cierre. A continuación se explican los objetivos de cada fase: (Rodríguez Sánchez, 2015)

- ✓ *Inicio*

Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

- ✓ *Ejecución*

En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

- ✓ *Cierre*

En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

### *1.5.2 Lenguaje de Modelado Unificado (UML).*

El Lenguaje de Modelado Unificado (UML: *Unified Modeling Language*) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos. El proceso indica los pasos que se deben seguir para llegar a un diseño. La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto informático. Si se quiere discutir un diseño con alguien más, ambos deben conocer el lenguaje de

modelado y no así el proceso que se siguió para obtenerlo. (Cornejo, 2008). Además, es un método de modelado, lo cual aporta las siguientes ventajas:

- ✓ Se puede usar para diferentes tipos de sistemas.
- ✓ Consolida muchas de las notaciones y conceptos más usados orientados a objetos.
- ✓ Mejor soporte a la planeación y al control de proyectos.
- ✓ Alta reutilización y minimización de costos.
- ✓ Fácil actualización o modificado del software a programar.

### *1.5.3 Herramientas CASE*[6]

Estas herramientas modelan la información de negocios cuando ésta se transfiere entre distintas entidades organizativas en el seno de una compañía. El objetivo primordial de las herramientas de esta categoría consiste en representar objetos de datos de negocios, sus relaciones y ayuda a comprender mejor la forma en que fluyen estos objetos de datos entre distintas zonas de negocio en el seno de la compañía. Estas herramientas proporcionan una ayuda importante cuando se diseñan nuevas estrategias para los sistemas de información y cuando los métodos y sistemas no satisfacen las necesidades de la organización. (Asensio, 2016)

**Visual Paradigm 8.0**

*Visual Paradigm* para UML *Standard Edition* (VP-UML SE) es una plataforma de modelado diseñada para apoyar a los arquitectos de sistemas, desarrolladores y diseñadores UML para acelerar el proceso de análisis y diseño de aplicaciones empresariales complejas ( Visual Paradigm International Ltd., 2010). Se utilizará esta herramienta para generar los artefactos ingenieriles a través del ciclo de vida de desarrollo del sistema a desarrollar. Entre sus principales características se encuentran:

- ✓ Disponibilidad en múltiples plataformas, lo cual facilita que las realizaciones de diagramas de modelado no dependan del sistema operativo que se tenga instalado en el equipo de trabajo.
- ✓ Uso de un lenguaje estándar que es común a todo el equipo de desarrollo facilitando la comunicación, permitiendo la realización de diagramas y modelos durante el proceso de desarrollo.
- ✓ Licencia: gratuita y comercial, fácil de instalar y actualizar.

[6] ***Computer Aided Software Engineering*** **(CASE**): Ingeniería de Software Asistida por Computación.

*1.5.3 Entorno de desarrollo integrado*

Un entorno de desarrollo integrado o en inglés *Integrated Development Environment*(IDE) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa 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 clásico tales como C++, Java, C# y Visual Basic. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. (Eslava Muñoz, 2013)

**Visual Studio Ultimate 2013**

Visual Studio 2013 es el entorno de desarrollo integrado de Microsoft. Esta herramienta permite a equipos de desarrollo diseñar y crear aplicaciones atractivas adaptadas a los nuevos sistemas operativos y los diferentes dispositivos. Visual Studio Ultimate 2013 es la oferta de desarrollo de equipo que abarca todo para permitir a las organizaciones cumplir con estas demandas. Las herramientas de descubrimiento y validación de la arquitectura permiten a los equipos mantener un alto grado de integridad arquitectónica y administrar eficazmente la deuda técnica. Las herramientas de prueba de calidad de servicio ayudan a validar de forma proactiva la escalabilidad de las aplicaciones y servicios de software para cumplir con los requisitos de escala de destino. La gestión de incidentes y la depuración de producción integran perfectamente a los equipos de desarrollo y operaciones en la liberación, monitoreo y mantenimiento de aplicaciones en la producción. (Microsoft, 2014)

*1.5.4 Lenguaje de Programación y Biblioteca*

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Cada lenguaje tiene sus instrucciones o enunciados verbales propios, que se combinan para formar los programas de cómputo. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. El lenguaje de programación tiene la capacidad de especificar, de forma precisa, cuáles son los datos que debe trabajar un equipo informático, de qué modo

deben ser conservados o transferidos dichos datos y qué instrucciones debe poner en marcha la computadora ante ciertas circunstancias. (Sebesta, 2012)

**C++**

C++ es un lenguaje de programación de propósito general diseñado para hacer la programación más agradable para el programador. C++ es un lenguaje orientado a objetos al que se le añadieron características y cualidades de las que carecía el lenguaje C. Las implementaciones de C++ existen desde algunos de los microordenadores más modestos hasta los superordenadores más grandes y para casi todos los sistemas operativos. (Stroustrup, 2013)

Para llevar a cabo la creación del componente de navegación en tiempo real sobre los flujos de video transmitidos por las cámaras IP se ha seleccionado a C++ como lenguaje de programación por su rapidez, portabilidad y documentación en el procesamiento de imágenes. La elección de este lenguaje se debe a las siguientes características:

- ✓ **Programación Orientada a Objetos:** Permite expresar programas en los términos de "objetos", que están destinados a modelar objetos en el mundo real. Tal paradigma permite que el código sea reutilizado de manera notable y sea más fácil de entender además de incrementar la productividad y calidad del software.
- ✓ **Eficiencia:** Es uno de los lenguajes más rápidos en cuanto a tiempo de ejecución ya que permite la separación de un programa en módulos que admiten compilación independiente, ventaja que puede ser aprovechada para el procesamiento de los flujos de video. Además, incluye un uso de apuntadores para la memoria, arreglos, estructuras y funciones, al mismo tiempo que es capaz de manejar actividades de bajo nivel y generar programas eficientes.

**Framework GStreamer 1.0**

GStreamer es un *framework* multimedia libre multiplataforma escrito en el lenguaje de programación C, usando la biblioteca GObject. GStreamer permite crear aplicaciones audiovisuales. Por ejemplo, con GStreamer se puede reproducir música o realizar tareas más complejas como mezclar audio y video. Este *framework* soporta de forma predeterminada codificación y decodificación en numerosos formatos, además puede soportar formatos adicionales a través de complementos. La función del núcleo de

GStreamer es proveer un *framework* para complementos, flujo de datos y manejo de distintos tipos de medios. También provee una API[7] para escribir aplicaciones. (Gstreamer, 2012)

**Características de GStreamer:**

- Una API para soportar dispositivos de cámaras, audio y video
- Permite mezclar diferentes tipos de medios como pueden ser audio, video, imágenes y subtítulos.
- Posibilita aplicar diferentes tipos de filtros, tanto de audio como de video.
- Facilita la transmisión y recepción de datos en tiempo real.
- Posibilita la reproducción de archivos multimedia.

Utilizando GStreamer se logra garantizar funcionalidades básicas y útiles del componente. También GStreamer provee un elemento que permite hacer *buffers*[8], además permite definir qué tiempo se desea almacenar el buffer, y tiene un elemento por el cual se puede especificar la posición sobre el buffer que se desea reproducir.

## Conclusiones

En este capítulo se precisaron una serie de conceptos asociados a la problemática planteada, donde se obtuvo una visión de los elementos y características a tener en cuenta durante el diseño y construcción del componente de navegación en tiempo real sobre flujos de video transmitidos por las cámaras IP. Además, se analizaron detalladamente las ventajas y características generales de las soluciones existentes a nivel nacional e internacional; arribándose a las siguientes conclusiones:

- ✓ El estudio de cómo se efectúan los procesos de navegación y captura de flujos de video provenientes de cámaras IP, posibilitó un mejor entendimiento sobre cómo se realizan los mismos en los sistemas de video vigilancia.
- ✓ La investigación y caracterización de las soluciones existentes a nivel nacional e internacional, reafirmó la necesidad del desarrollo de esta aplicación, al no existir una que resuelva la problemática planteada en la presente investigación. Además, permitió identificar el uso del *framework GStreamer* para la navegación sobre el flujo de video en tiempo real.

[7] **Application Programming Interface (API):** Interfaz de Programación de Aplicaciones.
[8] **Buffer:** Es un espacio de la memoria en un disco o en un instrumento digital reservado para el almacenamiento temporal de información digital, mientras que está esperando ser procesada.

- ✓ Las herramientas y tecnologías utilizadas en el desarrollo de la aplicación son las definidas por el sistema video vigilancia Suria, favoreciendo con esto la correcta integración entre el componente de navegación y el sistema de video vigilancia.

# Capítulo 2: Análisis y diseño de la solución propuesta

## Introducción

En el presente capítulo se realiza un estudio del dominio del negocio, que tiene como fin comprender las características que posee el mismo y que luego serán implementadas en la solución. Se identifican los requisitos funcionales y no funcionales, agrupándose en casos de uso y presentándose una descripción de cada uno de ellos. También se presentan los diagramas de clases del diseño y otros artefactos resultantes del flujo de trabajo de la metodología AUP-UCI, mostrándose un correcto uso de la arquitectura seleccionada y los patrones de diseño.

## 2.1 Modelo de dominio

Los modelos de dominio identifican las principales preocupaciones en el sistema. Estos se definen mediante diagramas de clases UML que incluyen objetos, atributos y asociaciones. (Sommerville, 2011). Se utiliza esta técnica debido a que no se tiene una estructura definida de los procesos del negocio, no se cuenta con un manual de procedimientos y los flujos de información son difusos.

El modelo de dominio permite:

- Identificar y representar conceptos del dominio de problema.
- Establecer y entender las relaciones entre los conceptos.

### *2.1.1 Descripción del flujo del diagrama del modelo de dominio*

El proceso de monitorización es iniciado por el módulo Visor, que se encuentra instalado en una estación de monitorización. Esta debe estar conectada a una red local para que pueda llevarse a cabo el proceso de visualización de los flujos de video emitidos por una o varias cámaras IP. El módulo Visor interactúa con el módulo Gestor, que es el encargado de gestionar los procesos del sistema Suria.

### *2.1.2 Diagrama del Modelo de dominio*

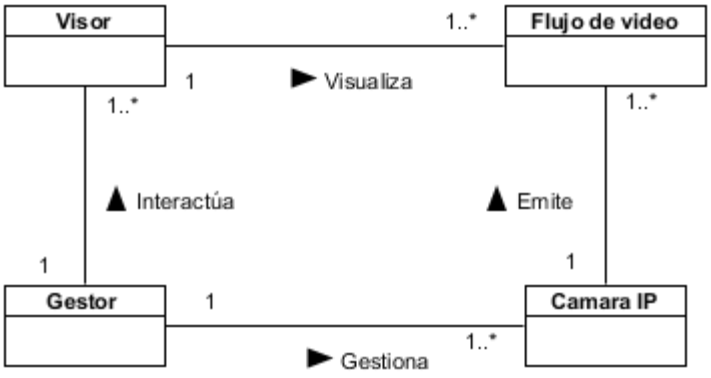


Fig. 1: Diagrama de clases del modelo de dominio.

*2.1.3 Descripción de las clases*

**Visor:** módulo del sistema Suria que permite la visualización de los flujos de video emitidos por las cámaras IP. Tiene la capacidad de reflejar todo el aspecto organizativo con que se manejan las cámaras internamente, además de poder visualizarlas de manera independiente o colectiva (a manera de vistas). También permite manipularlas de acuerdo a las capacidades de cada una.

**Flujos de Video:** representan la secuencia de imágenes que transmiten las cámaras IP. Se pueden visualizar desde el módulo Visor.

**Cámaras IP:** videocámaras diseñadas para enviar los flujos de video obtenidos de un área en específico a través de la red.

**Gestor:** módulo fundamental del sistema Suria, encargado de la interacción entre los demás módulos del sistema. Controla y supervisa todo el tráfico de información.

## 2.2 Especificación de los requisitos de software

La especificación de los requisitos de software no es más que la descripción completa del comportamiento del sistema que se desea desarrollar. La misma incluye los requisitos funcionales, que son capacidades o condiciones que el sistema debe cumplir, y los requisitos no funcionales o complementarios, que son propiedades o cualidades que el producto debe tener. (Sommerville, 2011)

*2.2.1 Requisitos Funcionales*

Los requisitos funcionales para un sistema describen lo que el sistema debe hacer. Estos requisitos dependen del tipo de software que se está desarrollando, de los usuarios del software y del enfoque

general adoptado por la organización al redactar los requisitos. Cuando se expresan como requisitos del usuario, los requisitos funcionales se describen generalmente de una manera abstracta que puede ser entendida por los usuarios del sistema. Sin embargo, los requisitos funcionales del sistema describen en detalle las funciones del sistema, sus entradas, salidas y excepciones. En algunos casos, los requisitos funcionales también pueden indicar explícitamente lo que el sistema no debe hacer (Sommerville, 2011). Estos requisitos son las características fundamentales del sistema y expresan la capacidad de acción del mismo.

Durante la especificación de requisitos se definieron 4 requisitos funcionales para el componente de navegación sobre los flujos de video transmitidos por las cámaras IP.

Nota: Se usa el prefijo RF en su nomenclatura.

**RF1:** Capturar flujo de video

**Descripción:** El sistema debe permitir capturar el flujo de video proveniente de una cámara IP.

**RF2:** Almacenar flujo de video.

**Descripción:** El sistema debe almacenar los flujos de video capturados en el disco duro de la estación de monitorización.

**RF3:** Navegar sobre el buffer.

**Descripción:** El sistema debe permitirle al usuario moverse hacia diferentes posiciones dentro del buffer.

**RF4:** Restablecer transmisión.

**Descripción:** El sistema debe permitirle al usuario regresar a la transmisión en vivo.

### *2.2.2 Requisitos No Funcionales*

Los requisitos no funcionales son las restricciones de los servicios o funciones ofrecidas por el sistema, restringen el espacio de posibles soluciones. Debe pensarse en estos, como las características que hacen al producto atractivo, usable, rápido o confiable (Pressman, 2010).

Nota: Se usa el prefijo RnF en su nomenclatura.

**Requisitos de Software**

**RnF1:** La solución deberá ser ejecutada en una computadora con los siguientes requisitos:

*Framework* multimedia *GStreamer* 1.0.

Sistema Operativo: Windows 7 o versiones superiores.

Microsoft.Net Framework 3.5.

**Requisitos de Hardware**

**RnF2:** La computadora empleada para el uso de la solución deberá disponer como mínimo de las siguientes prestaciones:

Procesador: Core i3 de 4ta a 3.30 GHz.

Memoria: RAM de 4 GB.

Almacenamiento: Disco Duro de 180 GB.

Conexión: Dispositivo de red con una velocidad de 100 Mbps.

## 2.3 Descripción del sistema

Luego de identificados los requisitos funcionales del componente de navegación sobre los flujos de video provenientes de cámaras IP para el sistema Suria, se agruparon en casos de uso (CU), los cuales se representan en el diagrama de casos de uso del sistema. Cada uno de los casos de usos identificados engloba un conjunto de acciones, las cuales son inicializadas por los actores del sistema.

*2.3.1 Definición de los actores*

Un actor del sistema no es más que un conjunto de roles que los usuarios desempeñan cuando interactúan con los casos de uso (Pressman, 2010). Los actores identificados para el componente de navegación son el proceso automático y el usuario del sistema de video vigilancia Suria.

Tabla 1: Actores del sistema.

| Actor | Descripción |
|---|---|
| **Proceso Automático** | Proceso que inicia automáticamente cada vez que se realiza una petición de reproducción de una cámara. Es el encargado de realizar e inicializar los CU Capturar y Almacenar flujos de |

| | |
|---|---|
| | video. |
| **Usuario** | Usuario del sistema de video vigilancia Suria. Es el encargado de realizar e inicializar los CU Navegar sobre el buffer y restablecer transmisión. |

*2.3.2 Listado de los Casos de Uso.*

Un caso de uso (CU) describe el comportamiento del sistema en distintas condiciones en las que el sistema responde a una petición de alguno de sus participantes. En esencia, un caso de uso narra una historia estilizada sobre como interactúa un usuario final (que tiene cierto número de roles posibles) con el sistema en circunstancias específicas. (Pressman, 2010) De acuerdo a los RF definidos anteriormente, se agruparon en los siguientes casos de usos:

**CU Capturar flujo de video**

RF1 Capturar flujo de video.

**CU Almacenar flujo de video**

RF2 Almacenar flujo de video.

**CU Navegar sobre el buffer.**

RF3 Navegar sobre el buffer.

**CU Restablecer transmisión.**

RF4 Restablecer transmisión.

*2.3.3 Diagrama de Casos de Uso del Sistema*

El diagrama de Casos de Uso del Sistema ayuda a determinar la funcionalidad y características del software desde la perspectiva del usuario, a través de una representación gráfica (Pressman, 2010). La siguiente figura muestra el diagrama de casos de uso del componente de navegación en tiempo real sobre los flujos de video provenientes de cámaras IP.

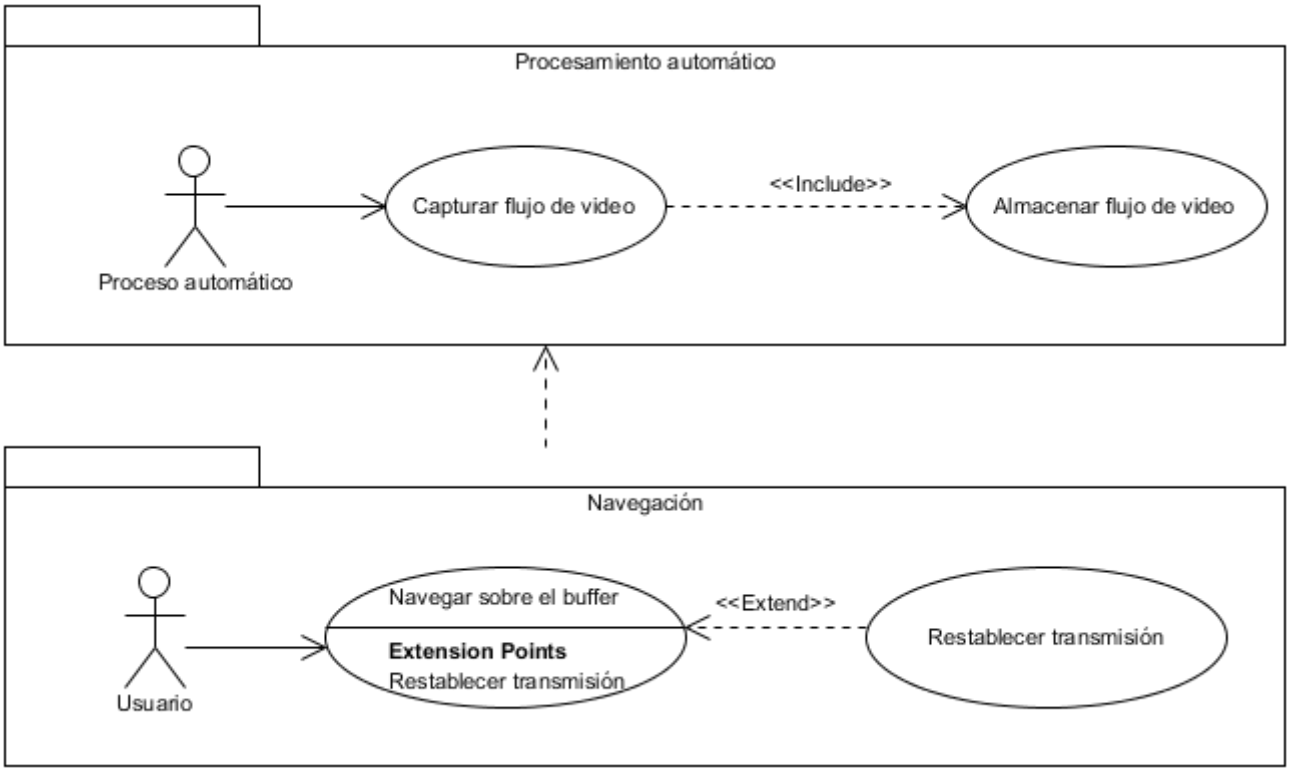


Fig. 2: Diagrama de Casos de Uso del Sistema

### *2.3.4 Especificación de los Casos de Uso del Sistema*

Una especificación de caso de uso es la versión más completa de un caso de uso, es donde se explica detalladamente lo que ocurre en los casos de uso del sistema. En el presente epígrafe se presenta la descripción textual del CU Navegar sobre el buffer.

Tabla 2: Descripción del CU Navegar sobre el buffer.

| | |
|---|---|
| **Objetivo** | El objetivo de este CU es permitir al usuario moverse por todo el video. |
| **Actores** | Usuario |
| **Resumen** | El caso de uso inicia de manera manual cuando se selecciona la cámara en la cual se quiere navegar. Una vez seleccionada la cámara esta muestra una barra de progreso, que permita al usuario situarse en cualquier posición del video. El CU termina cuando se regresa a la transmisión en vivo. |
| **Complejidad** | Alta. |
| **Prioridad** | Media. |

| | | |
|---|---|---|
| **Precondiciones** | Video del buffer.<br>Seleccionar una cámara. | |
| **Postcondiciones** | Barra de progreso sobre el video. | |
| **Flujo de eventos** | | |
| **Flujo básico:** Navegar sobre el buffer. | | |
| | **Actor** | **Sistema** |
| 1. | Seleccionar una cámara | |
| 2. | | Muestra una barra de progreso con la duración del buffer de video almacenado. |
| 3. | Navega sobre la barra de progreso, dando clic sobre la misma. | |
| | | Permite la navegación sobre el video. Lleva el video que se está reproduciendo hasta el instante de tiempo especificado. Termina el caso de uso. |
| **Flujos alternos** | | |
| **Nº** 1a. No se selecciona la cámara. | | |
| | **Actor** | **Sistema** |
| 1a | | El sistema no crea la barra de progreso. Termina el CU. |
| **Relaciones** | **CU incluidos** | No procede |
| | **CU extendidos** | Restablecer transmisión. |
| **Requisitos no funcionales** | RnF1, RnF2 | |
| **Asuntos pendientes** | | |
| **Prototipo elemental de interfaz gráfica de usuario** | | |
| | | |

## 2.4 Propuesta de solución

A continuación, se describe cómo funciona el componente de navegación:

1. Las cámaras IP envían flujos de video al servidor *streaming,* el cual transmite dichos flujos hacia los clientes que lo soliciten.
2. EL componente de navegación captura los flujos de video transmitidos por el servidor.

3. Al capturar los flujos de video se almacenan en un sector del disco duro de la estación de monitorización.
4. En el disco duro solo se van almacenando los últimos flujos de video de tal manera que le vaya mostrando al usuario los últimos 2 minutos transmitidos.
5. El usuario a través de una barra de progreso navega sobre esos flujos de video almacenados en el disco duro.
6. Al concluir la navegación se regresa a la transmisión en vivo y se eliminan los ficheros de tal manera que solo se queden almacenados los últimos 2 minutos de la transmisión.

## 2.5 Descripción de la Arquitectura

La arquitectura del software de un programa o sistema de cómputo es la estructura o estructuras del sistema, lo que comprende a los componentes del software, sus propiedades externas visibles y las relaciones entre ellos. (Pressman, 2010)

El sistema Suria establece una arquitectura Cliente-Servidor utilizando el estilo de llamada y retorno. Esta arquitectura es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. El cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio). El sistema cuenta con los módulos Visor, Administración y Recuperador, los cuales se ejecutan en la parte cliente de la solución y los módulos *Analitic*, *Recorder*, Autonomía, *ServerManager* se ejecutan en la parte del servidor. Cuando cada instancia del módulo Visor realiza peticiones de visualizar las cámaras IP, se envía una petición al Servidor que se encarga de establecer una conexión con la cámara solicitada, el cual captura el flujo de video y lo transmite de vuelta hacia el Visor.

La arquitectura cliente-servidor se rige de una arquitectura monolítica donde no hay distribución de tareas entre sí, solo una simple comunicación entre un usuario y una terminal en donde el cliente y el servidor no pueden cambiar de roles. Esta arquitectura monolítica consiste en considerar la tarea como un todo, es decir, estudiar simultáneamente el conjunto de aspectos a resolver e implementar un único código que realice todas las funciones necesarias. Esto trae como ventaja que funciona más rápido y es fácil de desarrollar. ( Liu, 2011)

El componente de navegación en tiempo real es de tipo *ClassLibrary*, lo cual es el mecanismo que implementa el *framework* .Net para tener un bajo acoplamiento entre sus componentes. La compilación del componente como una DLL (*Dinamic Link Library*) permite que sea utilizado por cualquier otra aplicación que implemente su interfaz.

En el componente de navegación las clases encargadas de todo el procesamiento de video fueron agrupadas en una sola capa lógica, por lo que fue desarrollado siguiendo la arquitectura monolítica. Es común en las aplicaciones desarrolladas en GStreamer el uso de la arquitectura monolítica, debido a que un software monolítico está diseñado para ser autónomo; es decir, los componentes del programa están independientes de otras aplicaciones informáticas en lugar de estar ligeramente acoplados a la misma. Los programas monolíticos suelen tener un mejor rendimiento y pueden ser más fáciles de probar y depurar porque, con menos elementos, hay menos variables que entran en juego ( Liu, 2011).

## 2.5 Patrones de diseño

Un patrón de diseño se caracteriza como una regla de tres partes que expresa una relación entre cierto contexto, un problema y una solución. Para el diseño de software, el contexto permite al lector entender el ambiente en el que reside el problema y qué solución sería apropiada en dicho ambiente. Un conjunto de requerimientos, incluidas limitaciones y restricciones, actúan como sistema de fuerzas que influyen en la manera en la que puede interpretarse el problema en este contexto y en cómo podría aplicarse con eficiencia la solución. (Pressman, 2010)

Los Patrones Generales de Software para Asignar Responsabilidades o *General Responsibility Assignment Software Patterns* (GRASP) son guías o principios que sirven para asignar responsabilidades a las clases (GAMMA, y otros, 2003). Teniendo en cuenta el patrón arquitectónico seleccionado, los patrones de diseño que se aplican en la implementación del componente de navegación son los siguientes:

**Controlador:** Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. (Larman, 2000)

**Experto:** Es el principio básico de la asignación de responsabilidades. Este patrón garantiza que cada clase cumpla con sus responsabilidades, según la información que contenga y las funcionalidades que se deseen implementar. (Larman, 2000)

**Alta Cohesión:** Una clase de diseño tiene un solo propósito: tiene un pequeño conjunto enfocado en responsabilidades y aplica atributos y métodos decisivos para implementar dichas responsabilidades. (Pressman, 2010)

**Bajo Acoplamiento:** Dentro del modelo de diseño es necesario que las clases de diseño colaboren unas con otras. No obstante, la colaboración debe mantenerse en un mínimo aceptable. Si un modelo de diseño está enormemente acoplado (todas las clases de diseño colaboran con todas las otras clases de diseño), el sistema es difícil de implementar, probar y mantener con el tiempo. En general, las clases de diseño dentro de un sistema deben tener solamente conocimiento limitado de otras clases. Esta restricción sugiere que un método sólo debe enviar mensajes a métodos en clases vecinas. (Pressman, 2010)

**Creador:** El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. Se aplica en todos los casos donde una clase tiene la responsabilidad de crear una nueva instancia de la otra (Larman, 2000). Se utilizó para decidir qué clase es la responsable de la construcción de objetos, es decir, la que le puede asignar los datos al constructor.

Los patrones creados por la Banda de los Cuatro o *Gang of Four* (**GoF**)[9], son agrupados en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. Los estructurales tratan la combinación de clases, su relación y la formación de estructuras de alta complejidad, mientras que los creacionales tratan la creación de instancias y los de comportamientos tratan la interacción y la cooperación entre clases. (GAMMA, y otros, 2003)

**Singleton:** Puesto en práctica con el fin de garantizar que una clase solo tenga una única instancia y proporcionar un punto de acceso global a ella. (Larman, 2005)

## 2.6 Modelo del diseño

El modelo de diseño es el conjunto de diagramas que describe el diseño lógico. Este ofrece una perspectiva de especificación o implementación, según lo desee el modelador. El modelo de diseño

[9] **GoF:** Nombre con el que se conoce comúnmente a los autores del libro *Design Patterns* (ISBN 0-201-63361-2), referencia en el campo del diseño orientado a objetos. La Banda de los Cuatro la componen los autores Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides.

perdurará durante todo el ciclo de vida del software y constituye la entrada fundamental utilizada para el correcto desarrollo de la implementación. (Larman, 2005)

*2.6.1 Diagrama de clases del diseño*

Un Diagrama de Clases del Diseño (DCD) permite modelar la vista de diseño del sistema y representar las asociaciones entre las clases y sus atributos. Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales; sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa. (Larman, 2005)

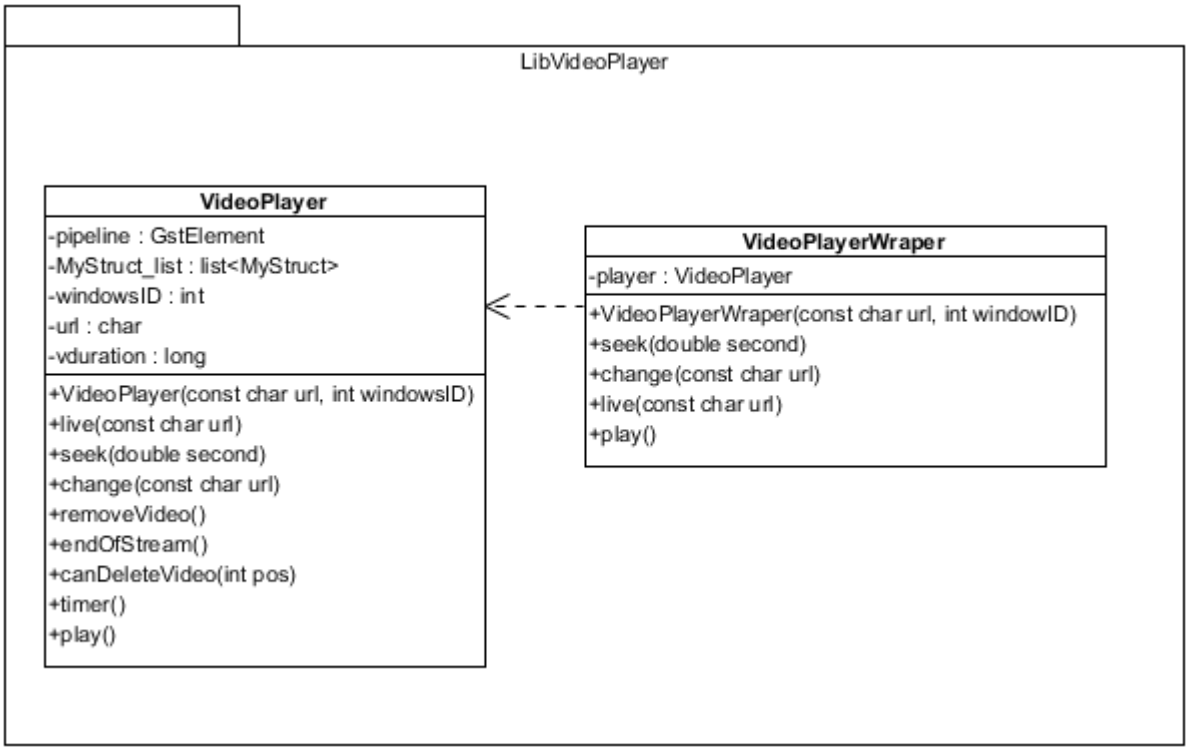


Fig. 3: Diagrama de Clases del Diseño

Como se observa en el diagrama anterior, la clase VideoPlayer es la encargada de la captura y almacenamiento de los flujos de video transmitidos por las cámaras IP. Además, también se encarga de navegar hacia el instante de tiempo especificado en los videos grabados para su posterior visualización. Por su parte la clase VideoPlayerWraper, permite llamar las funcionalidades nativas de C++ usadas en la clase VideoPlayer para posibilitar de esta forma la integración con un sistema. Esta clase es imprescindible para la integración del componente con el módulo Visor de Suria.

## Conclusiones

En este capítulo se especificaron los temas relacionados con el análisis y diseño de la aplicación que se desea desarrollar, a partir de la información recopilada con anterioridad:

- ✓ Se realizó una modelación de la situación problemática obteniéndose el modelo de dominio correspondiente con su respectivo diagrama, además de una descripción del flujo de dominio facilitando con esto una mejor comprensión sobre el funcionamiento del sistema, así como de las clases que interactúan dentro del mismo.
- ✓ Se identificaron los actores que interactúan con cada uno de los casos de usos, obteniéndose finalmente el diagrama de casos de uso del sistema, lo cual facilitó la realización de las especificaciones pertinentes, que permitirán al desarrollador conocer de forma detallada el flujo de cada una de las funcionalidades.
- ✓ Se estableció la línea base de la arquitectura que tendrá el componente de navegación que se quiere desarrollar empleándose la arquitectura monolítica, proporcionando con esto que se logre garantizar el mejor desempeño, portabilidad y flexibilidad del componente.
- ✓ Se identificaron los patrones de diseño GRASP utilizados en el desarrollo de la aplicación, permitiendo facilitar la asignación de responsabilidades logrando un diseño de software que sirva de apoyo a la implementación del componente.

# Capítulo3: Implementación y Prueba

## Introducción

En el presente capítulo se exponen todos los elementos relacionados al flujo de implementación y a las pruebas del sistema. También se describen las técnicas utilizadas en el desarrollo de la aplicación a través del modelo de despliegue, y se representa la aplicación en el diagrama de componentes. Por último, se describen las pruebas a las que fue sometida la aplicación, con el objetivo de comprobar su correcto funcionamiento.

## 3.1 Modelo de Implementación

Un modelo de implementación es un modelo para describir la implementación de los elementos del modelo de diseño en forma de componentes (ficheros de código, bibliotecas y ejecutables, entre otros). Permite representar las relaciones que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Este modelo describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizado, y cómo dependen los componentes unos de otros. (Jacobson, y otros, 2000)

### *3.1.1 Diagrama de Despliegue*

El Diagrama de Despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema y las relaciones existentes entre ellos. El diagrama está compuesto por nodos, dispositivos y conectores. El propósito del diagrama de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. (Larman, 2000). A continuación, se muestra el diagrama de despliegue correspondiente al componente de navegación el cual está compuesto por 3 nodos:

**Módulo Visor:** módulo del sistema de video vigilancia Suria que permite la visualización de los flujos de video emitidos por las cámaras IP. Este nodo además contiene el componente de navegación que permite el desplazamiento sobre los flujos de videos provenientes de la cámara IP.

**ServerManager:** servidor del sistema de video vigilancia Suria. Contiene los módulos *Recorder*, *Analitic* y Autonomía.

**Cámara IP:** dispositivo encargado de emitir los flujos de video.

A través del *módulo Visor* del sistema Suria, el usuario realiza peticiones de transmisión a las *cámaras IP* conectadas al sistema. Estas solicitudes son recibidas por el *ServerManager*, que emplea el protocolo RTSP para comunicarse con las cámaras y transmite el flujo de video capturado hacia el *módulo Visor* utilizando el mismo protocolo.

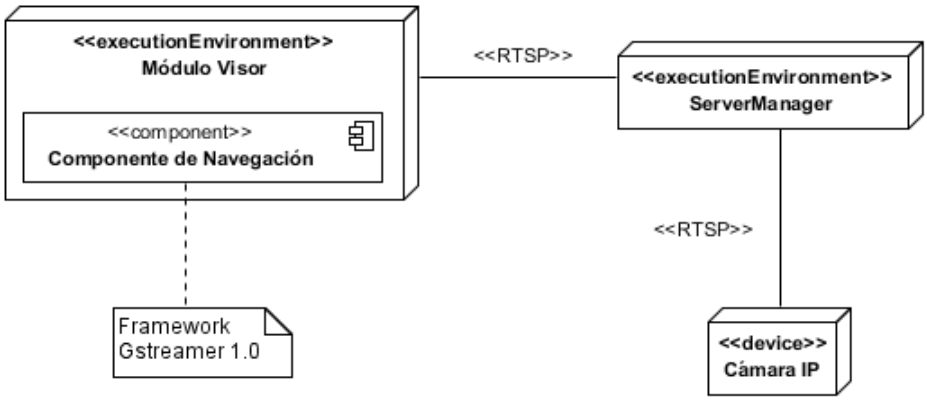


Fig. 4: Diagrama de Despliegue.

*3.1.2 Diagrama de componentes de implementación*

Los diagramas de componentes son la estructura del modelo de implementación. Este diagrama presenta paquetes donde se agrupan los elementos físicos de un sistema, así como las relaciones entre ellos. Los componentes representan todos los tipos de elementos de software que intervienen en la fabricación de aplicaciones informáticas. Estos pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente (Jacobson, y otros, 2000). A continuación, se muestra el diagrama de componentes perteneciente al componente de navegación:

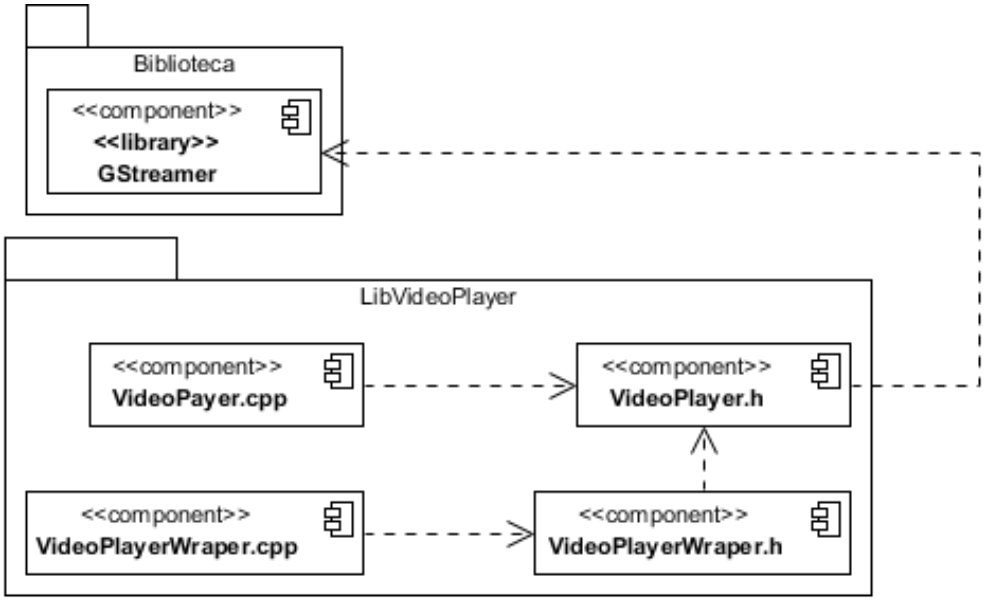


Fig. 5: Diagrama de Componentes de Implementación.

## 3.2 Estándar de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. A continuación, se presentan algunas de estas convenciones para la programación en lenguaje C++. (GitHub, 2017)

### *3.2.1 Estilos de codificación empleados*

**Comentarios:**

Cada programa deberá comenzar con un comentario que incluya:

- Autor
- Fecha
- Objetivo, o problema que resuelve el programa

Cada función debe tener un encabezado que contenga:

- Objetivo de la función y no descripción del procedimiento.
- Comentarios de apoyo a variables, llamadas a función o inclusión de archivos que no sean obvios al proceso.

- Explicación de uso de argumentos (parámetros) no obvios.
- Explicación de uso de valores devueltos (de retorno).

**Nombres de identificadores**

Se considera como identificador a los nombres de variables (arreglos, matrices, apuntadores), funciones, así como cualquier tipo de dato definido por el usuario (estructura, clase). Dichos identificadores deberán seguir las siguientes normas, además de las definidas por el propio lenguaje. (GitHub, 2017)

- Deberán tener un nombre significativo que permita su simple lectura, pueda conocerse su función, sin tener que consultar manuales o hacer demasiados comentarios.
- Para nombres que se usen con frecuencia o para términos largos, se recomienda usar abreviaturas estándar para que éstos tengan una longitud razonable. Si usa abreviaturas deben manejar la misma lógica en todo el programa.
- Evitar identificadores que comiencen con uno o dos caracteres de subrayado para evitar que se confundan con los que el compilador selecciona.

**Identificadores de variables**

Comenzarán siempre con la primera letra minúscula correspondiente a su tipo de dato. Para distinguir palabras dentro del nombre deberá emplearse una letra mayúscula o un guión bajo (_), sin mezclar ambas formas en un mismo programa (GitHub, 2017).

**Organización Visual del Programa**

- No manejar en los programas más de una instrucción por línea.
- Declarar las variables en líneas separadas.
- Añadir comentarios descriptivos junto a cada declaración de variables, si es necesario.
- Insertar una línea en blanco antes y después de una declaración de datos que aparezca entre instrucciones ejecutables.
- Las declaraciones de datos dentro de una función, deberán ir al inicio y separadas de las instrucciones ejecutables de la función por medio de una línea en blanco.
- Deben incluirse espacios en ambos lados de los operadores binarios.

## 3.3 Pruebas de Software

La calidad de un producto informático depende en gran medida de las pruebas de software que se le apliquen, ya que estas representan una revisión final de la codificación y del cumplimiento de las especificaciones del cliente, con el objetivo de encontrar anomalías y corregirlas antes de ser entregado al usuario final. (Pressman, 2010)

### *3.3.1 Prueba de Caja Blanca*

La prueba de caja blanca requiere una visión interna de cualquier producto sometido a ingeniería. Esta permite conocer el funcionamiento interno de un producto, pueden realizarse pruebas para garantizar que "todos los engranes embonan"; es decir, que las operaciones internas se realizan de acuerdo con las especificaciones y que todos los componentes internos se revisaron de manera adecuada. La prueba de caja blanca del software se basa en el examen cercano de los detalles de procedimiento. Las rutas lógicas a través del software y las colaboraciones entre los componentes se ponen a prueba al revisar conjuntos específicos de condiciones y/o bucles. (Pressman, 2010)

Para la aplicación de las pruebas de Caja Blanca se utilizó la técnica del Camino Básico que permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. (Pressman, 2010)

A continuación, se muestra en la figura el código, que pertenece a la funcionalidad de navegar

```
void VideoPlayer::seek(double second, int idcamera)
{
1   if (second>120)
    {
2    videoToNavigate(second, idcamera);
    }
   else
    {
     double test = second * 1000000000;
     gst_element_seek(pipeline, 1.0, GST_FORMAT_TIME,
3  GST_SEEK_FLAG_FLUSH,
   GST_SEEK_TYPE_SET, test,
   GST_SEEK_TYPE_NONE, GST_CLOCK_TIME_NONE);
    }
4  }
```

Fig. 6: Pruebas de Caja Blanca con la técnica del Camino Básico.

En la siguiente figura se muestra el grafo obtenido:

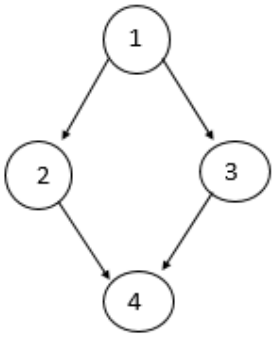


Fig. 7:Grafo obtenido del código seleccionado

Para el cálculo de la complejidad ciclomática se utiliza la siguiente fórmula V (G) = A – N + 2, donde [V (G)] es la complejidad ciclomática, [A] representa a la cantidad de aristas y [N] a la de nodos con los que cuenta el grafo de flujo asociado al código (Pressman, 2010). Luego de aplicar la fórmula para los valores del grafo, esta arrojó el siguiente resultado:

V (G) = A – N + 2

V (G) = 4 – 4 + 2

V (G) = 2

Debido a que el valor obtenido por complejidad ciclomática es 2, deja claro que existen 2 posibles caminos a recorrer para el flujo; de igual forma es la cantidad mínima de pruebas que se le realizó al código. La siguiente tabla recoge los caminos básicos del flujo:

Tabla 3 : Caminos básicos obtenidos del grafo.

| **No.** | **Caminos** |
|---|---|
| **1** | 1-2-4 |

| | |
|---|---|
| **2** | 1-3-4 |

A continuación, se pasa a la creación de casos de pruebas para el primer camino:

Tabla 4 Caso de prueba para el camino básico

| Caso de prueba para el camino básico No.1 | |
|---|---|
| Camino | 1-2-4 |
| Descripción de los parámetros de entrada | Recibe como parámetros una variable second que es la que contiene el tiempo a donde se va a empezar a reproducir el video, y una variable idcamera la cual se usará para saber qué cámara se está visualizando. |
| Resultado esperado | 1-2-4 |

*3.3.2 Prueba de Integración*

Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias y lo que prueban es que todos los elementos unitarios que componen el software, funcionan juntos correctamente probándolos en grupo. Las pruebas de integración exponen el estado incompleto o los errores de las especificaciones de la interfaz, debido a que se centra principalmente en probar la comunicación entre los componentes y sus comunicaciones ya sea hardware o software (Sommerville, 2005). Entre las técnicas de integración para realizar esta prueba se escogió el enfoque incremental.

Este enfoque permite a los desarrolladores integrar los módulos uno a uno utilizando trozos o controladores para descubrir los defectos. Además, como se prueba el programa en pequeñas porciones son más fáciles de detectar los fallos. En este enfoque existen dos métodos de prueba incrementales: Integración ascendente (*down-top*) e Integración descendente (*top-down*). En la presente investigación es utilizado el segundo método debido a que es la recomendada para integrar módulos generales. La técnica *top-down* comienza con los módulos de nivel superior, y se verifica que los módulos de nivel superior llaman a los de nivel inferior de manera correcta, con los parámetros correctos. Los módulos subordinados al módulo de control principal se van incorporando a la estructura, bien de forma primero en profundidad, o bien de forma primero en anchura. La integración primero en profundidad integra todos los módulos de un camino de control principal de la estructura. (Sommerville, 2005)

Para probar la correcta integración del componente de navegación y el módulo Visor de Suria se utiliza la integración incremental descendente. La ejecución de la prueba queda explicada en los siguientes pasos:

1. El módulo Visor de Suria envía una solicitud de transmisión de una cámara al *ServerManager*, el cual envía una dirección a la que conectarse para obtener el flujo de video.
2. El componente de navegación captura el flujo de video, y se mantiene almacenando un *buffer* con los dos últimos minutos que se han estado transmitiendo.
3. Cuando el usuario selecciona la opción Iniciar navegación, se comienza a mostrar el video almacenado. A medida que el usuario se desplace sobre la barra de progreso el componente de navegación lo conduce a ese instante de tiempo dentro del video. Se recibe el flujo de video correctamente en el módulo Visor.

### *3.3.3 Prueba de sistema*

La prueba del sistema abarca una serie de pruebas diferentes cuyo propósito principal es ejercitar intensamente el sistema de cómputo. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se hayan integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas. (Pressman, 2010) Este tipo de prueba es utilizado para comprobar la calidad del componente de navegación en cuanto a la escalabilidad, la fiabilidad y el uso de los recursos.

Las pruebas fueron realizadas en una estación de trabajo con las siguientes prestaciones:

Sistema Operativo: Windows 10

Procesador Core i3-2120 a 3.30 GHz.

Memoria RAM de 4 GB.

**Pruebas de Estrés:** pruebas para encontrar el volumen de datos o de tiempo en que la aplicación comienza a fallar o es incapaz de responder a las peticiones. Son pruebas de carga o rendimiento, pero superando los límites esperados en el ambiente de producción y determinados en las pruebas. (Lara García, 2013) Este tipo de prueba se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada. A continuación, se presentan los resultados de la prueba de estrés realizada, probando el funcionamiento del componente de navegación con más de 10 cámaras al mismo tiempo.

Tabla 5: Resultados de la Prueba de Estrés aplicada al componente de navegación.

| Cantidad de cámaras | CPU (%) | RAM (%) | Respuesta del Sistema |
|---|---|---|---|
| 12 | 75 | 50.4 | El sistema responde todas las peticiones pero deja de almacenar el video de algunas de las cámaras. |
| 14 | 90 | 80 | El sistema no reproduce el video almacenado. |
| 16 | - | - | El sistema colapsa. |

**Pruebas de rendimiento:** estas pruebas se realizan para determinar lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo. Permiten probar cuán rápida y eficiente puede ser la respuesta del sistema en un entorno controlado, que debe simular lo más cercano a la realidad, el entorno de producción donde se utilizará el software. (Lara García, 2013)

Esta prueba fue realizada en dos estaciones de trabajo con diferentes prestaciones con el objetivo de comprobar el funcionamiento y el consumo de recursos de la aplicación en entornos variados. Una de las estaciones de trabajo tiene las siguientes:
Características: Sistema Operativo: Windows 10
Procesador: Intel Core i3-2120 CPU 3.30GHz
Memoria RAM: 4 GB

En la siguiente tabla se muestran los resultados de la prueba en la primera estación de trabajo, observándose un correcto funcionamiento de la aplicación, así como un uso eficiente de los recursos del computador.

Tabla 6: Resultados de las Pruebas de Rendimiento aplicadas al componente de navegación.

| Cantidad de cámaras | RAM (MB) | CPU (%) |
|---|---|---|
| 1 | 102.9 | 5.2 |

| | | |
|---|---|---|
| 2 | 160.3 | 7.8 |
| 4 | 346.7 | 20 |
| 6 | 621.5 | 48.7 |
| 10 | 693.5 | 63.5 |

A continuación, se muestran los resultados de la prueba aplicada en la segunda estación de trabajo.

Características: Sistema Operativo: Windows 10

Procesador: Intel Core i7-3190 CPU 3.60GHz

Memoria RAM: 8 GB

Tabla 7: Resultados de las Pruebas de Rendimiento aplicadas al componente de navegación.

| **Cantidad de cámaras** | **RAM (MB)** | **CPU (%)** |
|---|---|---|
| 1 | 51.2 | 2.2 |
| 2 | 80.0 | 4.3 |
| 4 | 173.4 | 14 |
| 6 | 310.1 | 28.6 |
| 10 | 345.8 | 36.9 |

Las pruebas de rendimiento permitieron evaluar el correcto funcionamiento del componente de navegación en varios entornos de trabajo.

## Conclusiones

En este capítulo se especificaron los temas relacionados con los procesos de implementación y prueba del componente de navegación sobre los flujos de video provenientes de cámaras IP para el sistema Suria, arribándose a las siguientes conclusiones:

- ✓ Se realizaron los diagramas de despliegue y de componentes de implementación, los cuales proporcionan una vista de la implementación del sistema garantizando una descripción detallada de su estructura.
- ✓ A través de las pruebas seleccionadas se pudo validar el cumplimiento de los requisitos funcionales establecidos en la fase inicial del proceso de desarrollo del producto, confirmando el correcto funcionamiento del componente de navegación. Los resultados obtenidos durante la fase

de pruebas representaron un elemento fundamental para corregir las deficiencias encontradas y así dar por concluida la construcción de un componente de navegación sobre los flujos de video transmitidos en vivo por las cámaras IP.

## Conclusiones Generales

Luego de haberse realizado todas las tareas de la investigación se puede afirmar que se cumplieron satisfactoriamente las mismas, arribando a las siguientes conclusiones:

- ✓ La investigación y caracterización de las soluciones existentes permitió conocer que estas no resuelven la problemática planteada, reafirmando la necesidad del desarrollo del componente de navegación.
- ✓ Las herramientas y tecnologías utilizadas en el desarrollo del componente son las definidas por el sistema de video vigilancia Suria, favoreciendo con esto la integración entre el componente de navegación y el sistema de video vigilancia.
- ✓ La utilización del *framework* multimedia *GStreamer* posibilitó al componente de navegación el procesamiento de los flujos de video emitidos por las cámaras IP, permitiendo de esta forma la navegación sobre estos.
- ✓ La utilización de los patrones de diseño y la arquitectura permitió aplicar una guía para la construcción del componente.
- ✓ Las pruebas realizadas al componente de navegación desarrollado demuestran que este cumple con las funcionalidades exigidas y está apto para ser integrado al sistema de video vigilancia Suria.

## Recomendaciones

Para mejorar el resultado y el impacto de la investigación se recomienda:

- ✓ Desarrollar en el módulo Visor una interfaz gráfica más intuitiva para interactuar con el componente de navegación.
- ✓ Permitir que los videos se vayan almacenando desde que se inicia el módulo Visor de Suria para todas las cámaras disponibles, y no que el almacenamiento comience al reproducir cada cámara.

## Referencias Bibliográficas

**Liu, Henry. 2011.** *Software Performance and Scalability: A Quantitative Approach.* New York, : NY John Wiley & Sons, 2011. ISBN: 9781118211311 1118211316.

**Mateos Costilla, David and Reaño Montoro, Samuel . 2008.** *Streaming de Audio/Video. Protocolo RTSP.* 2008.

**Visual Paradigm International Ltd. 2010.** Visual Paradigm for UML Standard 8.0 . *Soft 112.* [Online] agosto 16, 2010. [Cited: noviembre 19, 2016.] http://visual-paradigm-for-uml-standard.soft112.com/.

**Asensio, Rafael Menéndez-Barzanallana. 2016.** Ingeniería del software. [Online] septiembre 9, 2016. [Cited: noviembre 19, 2016.] http://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html.

**Camaraip emp. 2009.** ¿Cómo funciona una camara ip? [Online] 2009. [Cited: noviembre 2, 2016.] http://www.ipcamaras.com.uy/funcionamiento.php.

**CCM Benchmark group. 2016.** Introducción al video digital. *CCM.* [Online] noviembre 2016. [Cited: noviembre 22, 2016.] http://es.ccm.net/contents/738-introduccion-al-video-digital.

**Cornejo, José Enrique González. 2008.** *DocIRS.* [Online] Enero 2008. [Cited: Noviembre 15, 2016.] http://www.docirs.com/uml.htm.

**DATYS. 2016.** *Plataforma de video vigilancia Xyma Safe Vision.* La Habana : s.n., 2016.

**ESCALANTE, LAIN JARDIEL CÁRDENAS. 2015.** Maestro del Software. [Online] enero 27, 2015. [Cited: diciembre 13, 2016.] http://maestrodelsoftware.blogspot.com/2015/01/9-el-modelo-de-dominio.html.

**Eslava Muñoz, Vicente Javier. 2013.** *Aprendiendo a programar paso a paso con C.* Madrid : Bubok Publishing S.L., 2013. ISBN: 9788468610627 8468610623.

**ESYS. 2016.** *LA VIDEOVIGILANCIA EN LA SEGURIDAD Análisis y recomendaciones para su actualización legal .* 2016.

**GALLARDO, SERGIO ANTONIO PINO. 2008.** ARQUITECTURA EN CAPAS PARA EL DESARROLLO. [Online] Diciembre 23, 2008. [Cited: Marzo 5, 2017.] http://revistas.uis.edu.co/.

**Gallego, José Carlos and Folgado, Laura. 2011.** *Periféricos de entrada (Montaje y mantenimiento de equipos).* 2011. ISBN: 978-84-9003-018-9.

**GAMMA, ERICH, y otros. 2003.** *Patrones de Disefio.* Madrid : PEARSON EDUCACION, 2003.

**García Mata, Javier. 2011.** *Videovigilancia: CCTV usando vídeos IP.* s.l. : Vértice, 2011. ISBN: 978-84-9931-356-6.

**GitHub. 2017.** https://github.com. *https://github.com.* [En línea] 2017. [Citado el: 4 de Marzo de 2017.] https://google.github.io/styleguide/cppguide.html.

**GNOME. 2008.** GStreamer. *GNOME Developer.* [Online] Mayo 2008. [Cited: novienbre 19, 2016.] https://developer.gnome.org/platform-overview/stable/tech-gstreamer.html.es.

**Gomez Bastar, Sergio. 2012.** *METODOLOGÍA DE LA INVESTIGACIÓN.* México : s.n., 2012.

**Gstreamer. 2012.** Gstreamer open source multimedia framework. [Online] 2012. [Cited: junio 2, 2017.] https://gstreamer.freedesktop.org/features/index.html.

**Howard. 2010.** *Application Note Time-Shift recording .* 2010.

**INTPLUS. 2017.** VIDEOVIGILANCIA.com. *VideoVigilancia.com Expertos en Seguridad.* [Online] Abril 19, 2017. [Cited: Junio 8, 2017.] http://videovigilancia.com/respvideovigilancia.htm.

**Ivideon. 2016.** *Ivideon. Videovigilancia en la nube.* [Online] 2016. [Cited: Octubre 26, 2016.] https://es.ivideon.com/.

**Jacobson, Ivar, Booch, Grady and Rumbaugh, James. 2000.** *El Proceso Unificado de Desarrollo de Software.* Madrid : 84-7829-036-2, 2000.

**Jean-François Pillou, Israel Ayala, Ángel Ortega, Carlos Villagómez , Carlos Villagómez. 2016.** Lenguajes de programación. *CCM.* [Online] Noviembre 2016. [Cited: noviembre 19, 2016.] http://es.ccm.net/contents/304-lenguajes-de-programacion.

**Kruegle, Herman. 2007.** *CCTV Surveillance: Video Practices and Technology, second edition.* s.l. : Burlington, MA, 2007. ISBN-13: 978-0-7506-7768-4 .

**Ksenos. 2016.** *KSENOS.* [Online] 2016. [Cited: Octubre 26, 2016.] https://ksenos.com/.

**Kumar, Vikas and Svensson, Jakob. 2015.** *Promoting Social Change and Democracy Through Information Technology.* s.l. : Hershey, PA, 2015. ISSN: 2326-9103.

**Lara García, Marisol. 2013.** *Pruebas de Software.* México : s.n., 2013.

**Larman, Craig. 2005.** *Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design and Iterative Development.* 2005.

**—. 2000.** *UML y Patrones.* Mexico : s.n., 2000. ISBN: 970-17-0261-1.

**Meissner, Michael. 2005.** *¿Analógico o digital?* Kiel : s.n., 2005.

**MEPSD. 2008.** Conceptos básicos de vídeo. *Diseño de Materiales Multimedia.* [Online] 2008. [Cited: Octubre 26, 2016.] http://www.ite.educacion.es/formacion/materiales/107/cd/video/video0101.html.

**Microsoft. 2014.** Microsoft. [Online] noviembre 7, 2014. [Cited: junio 2, 2017.] https://www.microsoft.com/en-us/.

**NCH Software . 2015.** Eyeline, software para videovigilancia. [Online] 2015. [Cited: Octubre 21, 2016.] http://www.nchsoftware.com/surveillance/es/.

**Novoa, Pablo Montero. 2007.** *Servidor modular de streaming.* Madrid : s.n., 2007.

**Ordinas, José María Barceló. 2008.** *Protocolos y aplicaciones Internet.* Barcelona : UOC, 2008.

**Pressman, Roger S. 2010.** *Ingeniería del software.* New York : Higher Education, 2010. ISBN: 978-607-15-0314-5.

**Rodríguez Sánchez, Tamara . 2015.** *Metodología de desarrollo para la.* La Habana : s.n., 2015.

**Rosenfeld, Azriel, Doermann, David and DeMenthon, Daniel. 2003.** *Video Mining.* 2003. ISBN: 978-1-4419-5383-4.

**Schulzrinne, Henning, Rao, Anup and Lanphier, Robert. 1998.** *Real Time Streaming Protocol (RTSP).* California : RFC 1889, 1998.

**Sebesta, Robert W. 2012.** *Concepts of programming languages (10th Edition).* 2012. ISBN 13: 978-0-13-139531-2.

**Solano, Fernando. 2012.** *Reproductor de Windows Media.* 2012.

**Sommerville, Ian. 2005.** INGENIERÍA DEL SOFTWARE. Séptima Edición. *INGENIERÍA DEL SOFTWARE. Séptima Edición.* Madrid : ISBN: 84-7829-074-5, 2005.

**—. 2011.** *SOFTWARE ENGINEERING.* United States of America : s.n., 2011.

**—. 2010.** *Software Engineering, 9th edition.* 2010.

**Soto Villalobos, Miguel. 2017.** sdpl60 Manual de Usuario Caja Decodificadora TV Digital Terrestre. [Online] Enero 2017. [Cited: Mayo 2, 2017.] http://docplayer.es/31856928-Sdpl60-manual-de-usuario-gaja-decodificadora-tv-digital-terrestre.html.

**Stroustrup, Bjarne. 2013.** *The C++ programming language (4th Edition).* s.l. : Pearson Education, 2013. ISBN-13: 978-0-321-56384-2.

**UC3M. 2012.** Protocolos de Transporte tutorial sobre UDP y TCP. *Universidad Carlos III de Madrid - Departamento de Ingeniería Telemática.* [Online] Febrero 25, 2012. [Cited: Noviembre 8, 2016.] http://www.it.uc3m.es/lpgonzal/protocolos/transporte.php.

**Velasco, Rubén. 2014.** iSpy, el software de video vigilancia OpenSource más completo. *iSpy.* [Online] 2014. [Cited: Octubre 21, 2016.] http://www.redeszone.net/2014/09/13/ispy-el-software-de-videovigilancia-opensource-mas-completo/.

## Bibliografía

**Álvarez, Miguel Angel. 2001.** *¿Qué es Streaming?* San Juan : s.n., 2001.

**Forouzan, Behrouz A. 2008.** *Data Communications AND Networking Fifth Edition.* New York : McGraw-Hill, 2008. ISBN 978-0-07-337622-6.

**GAMMA, ERICH, y otros. 2003.** *Patrones de Disefio.* Madrid : PEARSON EDUCACION, 2003.

**Larman, Craig. 2003.** *Modelo del Dominio.* Burnaby : s.n., 2003.

**Larman, Craig. 2000.** *UML y Patrones.* Burnaby : s.n., 2000.

**Pressman, Roger S. 2010.** *"Software Engineering".* 7ma. New York : Higher Education, 2010.

**QtGStreamer. 2014.** QtGStreamer. [En línea] 8 de Julio de 2014. http://gstreamer.freedesktop.org.

**Sommerville, Ian. 2005.** *Ingeniería de Software.Séptima Edición.* Madrid : Pearson Education, 2005. ISBN: 84-7829-074-5.

**Stroustrup, Bjarne. 2013.** *The C++ programming language (4th Edition). s.l. : Pearson Education, 2013. ISBN-13: 978-0-321-56384-2.*

**Taymans, Wim , y otros. 2012.** *GStreamer Application Development.* Barcelona : s.n., 2012.

# Anexos

## I. Entrevista realizada

**Fecha:** noviembre 2014

**Entrevistados:** Reidel Morales Toledo, Solangel Rodríguez Vázquez y Rolando Alberto Escobar Casanova

**Preguntas:**

1. ¿Considera usted importante el desarrollo de un componente de navegación sobre los flujos de video transmitidos por las cámaras IP? ¿Qué beneficios traería para el sistema de video vigilancia Suria un componente de este tipo?
2. ¿Qué requisitos considera que sean importantes a desplegar por el componente de navegación?

**Resultados generales de la entrevista por preguntas:**

1. Una componente de navegación sobre los flujos de video transmitidos por las cámaras IP provee al sistema de video vigilancia Suria de un entorno más seguro, permitiendo observar instantes anteriores que no han sido grabados. Además, traería como beneficio para el sistema de video vigilancia Suria un componente capaz de permitirle al usuario retroceder la transmisión en vivo, para analizar con más detalle el video en caso de que este desatendiera la pantalla.
2. Algunos de los requisitos coincidentes mencionados por los entrevistados son:
   - Capturar flujo de video.
   - Almacenar flujo de video.
   - Navegar sobre el buffer.
   - Restablecer transmisión.

## II. Descripción de los Casos de Uso del Sistema

Tabla 8: Descripción del CU Capturar flujo de video.

| | |
|---|---|
| **Objetivo** | El objetivo de este CU es permitir capturar el flujo de video proveniente de una cámara IP. |
| **Actores** | Proceso automático. |

| | |
|---|---|
| **Resumen** | El caso de uso inicia cuando el Visor recibe la solicitud de transmisión de una cámara IP. Una vez hecha la petición se capturan los flujos emitidos por el dispositivo de seguridad. El CU termina cuando la cámara deja de transmitirse. |
| **Complejidad** | Media. |
| **Prioridad** | Alta. |
| **Precondiciones** | Dirección URL de la cámara IP (Cadena de caracteres). |
| **Postcondiciones** | Flujo de video emitido por la cámara IP. |

| **Flujo de eventos** | | |
|---|---|---|
| **Flujo básico:** Capturar flujo de video | | |
| | **Actor** | **Sistema** |
| 1. | Realiza una petición con el identificador de la cámara a la cual se desea guardar el buffer. | |
| 2. | | Recibe la dirección URL de la cámara IP que se comenzó a transmitir. |
| 3. | | Solicita al servidor el flujo de video de la cámara solicitada. Termina el caso de uso. |
| **Flujos alternos** | | |
| **Nº** 1a. Cámara IP apagada. | | |
| | **Actor** | **Sistema** |
| 1a | | La cámara IP se encuentra apagada y no devuelve flujo de video. En su lugar se mostrará una pantalla en negro. |
| **Flujos alternos** | | |
| **Nº** 3a. No se puede capturar el flujo de video. | | |
| | **Actor** | **Sistema** |
| 3a | | El sistema no recibe la dirección URL de la cámara IP debido a que estas están apagadas. Al no recibir ninguna dirección URL de las cámaras IP el sistema no inicia la captura del flujo. Termina el CU. |

| | | |
|---|---|---|
| **Relaciones** | **CU incluidos** | Almacenar flujo de video. Ver CU Almacenar flujo de video |
| | **CU extendidos** | No procede. |
| **Requisitos no funcionales** | RnF1, RnF2 | |
| **Asuntos pendientes** | | |
| **Prototipo elemental de interfaz gráfica de usuario** | | |
| | | |

Tabla 9: Descripción del CU Almacenar flujo de video

| | | |
|---|---|---|
| **Objetivo** | El objetivo de este CU es permitir almacenar el flujo de video proveniente de una cámara IP. | |
| **Actores** | Proceso automático. | |
| **Resumen** | El caso de uso inicia de manera automática cuando son capturados los flujos de videos emitidos por la cámara IP. Una vez capturado los flujos estos se guardan dentro de un fichero en el disco duro. El CU termina cuando la cámara deja de transmitirse. | |
| **Complejidad** | Alta. | |
| **Prioridad** | Alta. | |
| **Precondiciones** | Flujo de video emitido por la cámara IP. | |
| **Postcondiciones** | Fichero con el flujo de video almacenado. | |
| **Flujo de eventos** | | |
| **Flujo básico:** Almacenar el flujo de video | | |
| | **Actor** | **Sistema** |
| 1. | | Inicia cuando los flujos de videos emitidos por la cámara IP son capturados. |
| 2. | | Recibe los flujos de videos provenientes de la cámara IP. |
| 3. | | Guarda dentro de un fichero en el disco duro los flujos de videos recibidos. Termina el caso de uso. |
| **Flujos alternos** | | |

| **Nº** 1a. No se puede capturar el flujo de video. | | | |
|---|---|---|---|
| | **Actor** | | **Sistema** |
| 1a | | | El sistema no captura los flujos de video. Al no recibir ningún flujo de video el sistema no puede almacenar. Termina el CU. |
| **Relaciones** | **CU incluidos** | No procede | |
| | **CU extendidos** | No procede | |
| **Requisitos no funcionales** | RnF1, RnF2 | | |
| **Asuntos pendientes** | | | |
| **Prototipo elemental de interfaz gráfica de usuario** | | | |
| | | | |

Tabla 10: Descripción del CU Restablecer transmisión

| **Objetivo** | El objetivo de este CU es volver a mostrar la transmisión en vivo. | |
|---|---|---|
| **Actores** | Usuario | |
| **Resumen** | El caso de uso inicia de manera manual cuando termina la navegación sobre el buffer. A la vez que este termina se muestra la transmisión en vivo. El CU termina cuando se muestra la transmisión en vivo. | |
| **Complejidad** | Baja. | |
| **Prioridad** | Media. | |
| **Precondiciones** | Barra de progreso sobre el video. | |
| **Postcondiciones** | Transmisión en vivo. | |
| **Flujo de eventos** | | |
| **Flujo básico:** Regresar a la transmisión en vivo | | |
| | **Actor** | **Sistema** |
| 1. | Termina la navegación sobre el buffer | |
| | | Se detiene la reproducción que se está llevando a cabo desde el archivo grabado. |
| 2. | | Se muestra los flujos de video en vivo capturados por la cámara. |
| **Relaciones** | **CU incluidos** | No procede |
| | **CU extendidos** | No procede. |

| | |
|---|---|
| **Requisitos no funcionales** | RnF1, RnF2 |
| **Asuntos pendientes** | |
| **Prototipo elemental de interfaz gráfica de usuario** | |
| | |

## Glosario de Términos

**Random Access Memory (RAM):** Memoria de acceso aleatorio.

**Computer Aided Software Engineering (CASE):** Ingeniería de Software Asistida por Computación.

**Application Programming Interface (API):** Interfaz de Programación de Aplicaciones.

**Ministerio de Educación, Política Social y Deporte(MEPSD):** Departamento ministerial con competencias en educación, asuntos sociales y deporte.

**Universidad Carlos III de Madrid (UC3M):** Universidad pública española situada en la Comunidad de Madrid.

**Real Time Streaming Protocol (RTSP):** El protocolo de transmisión en tiempo real (del inglés *Real Time Streaming Protocol*) establece y controla uno o muchos flujos sincronizados de datos, ya sean de audio o de video. El RTSP actúa como un mando a distancia mediante la red para servidores multimedia.

**User Datagram Protocol (UDP):** Es un protocolo del nivel de transporte basado en el intercambio de datagramas.

**Transmission Control Protocol (TCP):** El Protocolo de Control de Transmisión es usado para crear "conexiones" entre computadoras a través de las cuales puede enviarse un flujo de datos.

**Asymmetric Digital Subscriber Line (ADSL):** Es una tecnología de acceso a Internet de banda ancha.

**Circuitos Cerrados de Televisión (CCTV):** Es una tecnología de video vigilancia diseñada para supervisar una diversidad de ambientes y actividades.