UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

Título: Componente para la evaluación de la calidad en las transmisiones de video basado en la técnica Sin Referencia.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor

Roberto Miguel Salas Armas

Tutores

Ing. Janet Cristina Labrada Paneque

Ing. Mónica Vigil Martínez

Ing. Guillermo Luzua Farias

La Habana, junio de 2015

"Año 57 de la Revolución"

Declaración de autoría

Declaro que soy el único autor del trabajo **"Componente para la evaluación de la calidad en las transmisiones de video basado en la técnica Sin Referencia"**, y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____

Autor: Roberto Miguel Salas Armas

_____ _____

Tutor: Ing. Janet Cristina Labrada Paneque **Tutor:** Ing. Mónica Vigil López

_____

Tutor: Ing. Guillermo Luzua Farias

Datos de contacto

Tutor: Ing. Janet Cristina Labrada Paneque

Formación académica:

Ingeniero en Ciencias Informáticas, Universidad de Ciencias Informáticas, 2012.

Centro Laboral:

Centro de Desarrollo de Geoinformática y Señales Digitales. Facultad 6.

Correo electrónico: janetcristina@uci.cu

Tutor: Ing. Mónica Vigil López

Formación académica:

Ingeniero en Ciencias Informáticas, Universidad de Ciencias Informáticas, 2012.

Centro Laboral:

Centro de Desarrollo de Geoinformática y Señales Digitales. Facultad 6.

Correo electrónico: mvigil@uci.cu

Tutor: Ing. Guillermo Luzua Farias

Formación académica:

Ingeniero en Ciencias Informáticas, Universidad de Ciencias Informáticas, 2012.

Centro Laboral:

Centro de Desarrollo de Geoinformática y Señales Digitales. Facultad 6.

Correo electrónico: gluzua@uci.cu

Resumen

La evaluación de la calidad de video digital es uno de los principales factores en el éxito de un sistema multimedia. Cuando un archivo de video es transmitido a través de redes de comunicación, puede degradarse, siendo necesario su control a tiempo para que la información que reciba el usuario resulte aceptable. El Sistema de Transmisión de Canales Virtuales entre sus funciones principales, se encarga de administrar canales de televisión y automatizar sus procesos, para garantizar calidad en la información que se transmite. Actualmente no cuenta con un mecanismo que le permita evaluar la calidad de video de las transmisiones en tiempo real que conviven en su entorno. El presente trabajo de diploma contiene la investigación y el proceso de desarrollo para obtener un componente de evaluación de calidad de video utilizando la técnica Sin Referencia, que contribuya a la monitorización de las trasmisiones del Sistema de Transmisión de Canales Virtuales del centro de desarrollo Geoinformática y Señales Digitales. El empleo de la técnica Sin Referencia le confiere independencia a la evaluación de la calidad, pues no necesita una referencia del video que se transmite para realizar la evaluación. Con el empleo de OpenCV como biblioteca para el procesamiento de imágenes, se analizaron las características de video digital: brillo, color, contraste y saturación para la evaluación de la calidad. Se realizaron experimentos que validaron la selección de los umbrales sobre dichas características, así como pruebas de software que ofrecieron resultados satisfactorios, demostrando la validez de la investigación realizada.

Palabras claves: evaluación, calidad, transmisión, Sin Referencia, video digital.

Abstract

The evaluation of digital video quality is one of the main factors in the success of a multimedia system. When a video file is sent through communication networks, can degrade it, being necessary its control on time, for that information that user receives be acceptable. The Broadcasting System Virtual Channels, between its main functions, manages television channels and automates its processes, to guarantee quality of information broadcasted. Nowadays, the Broadcasting System Virtual Channels does not have a mechanism that allows it to evaluate the video quality of broadcastings in real time that live in their environment. The present diploma work contains the investigation and development processes to get a video quality evaluation using No-Reference model, which contributes to monitoring of Broadcasting System Virtual Channels broadcastings of Geoinformatics and Digital Signals center. The use of No-Reference model confers it independence to video quality evaluation, which does not need the video broadcasted to emit an evaluation. With the use of OpenCV library for image processing, were analyzed the digital video characteristics: brightness, color, contrast and saturation for video quality evaluation. Experiments were realized to validate the selection of the thresholds over said characteristics, as well as software tests that offered successful results, demonstrating the validity of investigation realized.

Keywords: *evaluation, quality, broadcasting, No-Reference, digital video*.

Índice

Índice de tablas

Índice de figuras

Introducción

La comunicación es uno de los procesos fundamentales realizados por el hombre en la actualidad. Desde el surgimiento de la especie humana comenzaron a evidenciarse diferentes formas de expresar información, desde los métodos rudimentarios y arcaicos como los jeroglíficos, hasta llegar a la invención de mecanismos avanzados como la telefonía, la radio y la televisión. Esta última se ha convertido en una de las vías más utilizadas para la difusión masiva de la información, su avance viene acompañado del auge de la ciencia y las tecnologías. Actualmente, a través de este medio se transmite todo tipo de información, ya sea texto, audio o video.

Aún con el surgimiento de la televisión, no se contaba con un mecanismo que posibilitara gestionar la información que se mostraba a través de la pantalla, debido a que las transmisiones se realizaban en vivo, y lo que llegaba a los televidentes no tenía forma de ser anteriormente editado (Zielinski, 1999). Con el desarrollo del campo de la informática aparece el video digital, que es un tipo de grabación que hace referencia a la captación, procesamiento, transmisión y reconstrucción de una secuencia de imágenes y sonidos, que representan escenas en movimiento. El video digital dio vida a los sistemas de gestión de materiales audiovisuales, capaces de hacer procesamiento digital sobre este tipo de archivo, con el fin de mejorar su visualización y transmisión (Montagu, 2005).

El realizar procesamiento digital a una señal de video consiste en un conjunto de técnicas que se aplican a las señales, con el objetivo de mejorar su calidad y propagación en el medio. Al procesar un archivo de video, muchas veces se corre el riesgo de perder parte de su información en medio de todas las operaciones realizadas. El video digital está sujeto a varios tipos de degradaciones a través de los procesos de adquisición, compresión, almacenamiento, transmisión y reproducción (Agustí, y otros, 2010). Existen técnicas de compresión de video con pérdidas, que son las más usadas al reducir la capacidad necesaria en el almacenamiento o transmisión de los datos de video, éstas pueden modificar el archivo durante la cuantificación[1]. Además, en ocasiones el video que se transmite se encuentra con fallas de origen que, aunque el sistema por el que se transmite le cause la menor cantidad de degradaciones posible, el mismo se muestra con dificultades (Casar, 2005).

[1] Es un paso imprescindible en el proceso de digitalización de una señal analógica (Rao, y otros, 1996).

Cuando se habla de degradaciones en un video, se refiere a la alteración en los valores de sus características fundamentales. Los flujos de bits[2] de video enviados a través de canales que introducen errores, como canales inalámbricos, pueden no ser recibidos correctamente. A su vez, las redes basadas en conmutación de paquetes[3], pueden causar pérdidas o retrasos considerables de los paquetes recibidos, dependiendo de las condiciones de la red. Todos estos errores pueden traducirse en degradaciones en el video recibido (González, y otros, 2004).

La calidad de un video puede ser una propiedad que no se pueda apreciar a simple vista, en ocasiones se transmiten archivos multimedia que poseen baja calidad, sin embargo se muestran adecuadamente según la percepción humana. Existen dos tipos de métodos que permiten evaluar la calidad de video: subjetivos y objetivos. Las evaluaciones subjetivas son siempre los sistemas de medida más confiables, ya que en dichas evaluaciones se toman directamente la opinión de los usuarios. Según (Joskowicz, y otros, 2012) el método subjetivo es "*La manera más confiable de medir la calidad de una imagen o un video* (…)". Este método se basa en la percepción del usuario que está visualizando el flujo de video.

A su vez, existen métodos objetivos que permiten identificar de forma automática si el video digital que se transmite, se muestra con la calidad óptima para los televidentes, teniendo en cuenta además, las particularidades de este tipo de archivo no percibidas por la visión de una persona. Estos métodos objetivos desarrollan modelos, cuyas diferencias radican en la disponibilidad o no del video original que se transmite. Los modelos del tipo Sin Referencia (SR) estiman la calidad basándose únicamente en el análisis de la señal recibida, resultando éstos los más complejos de implementar. Los seres humanos no necesitan señales de referencia, ni información adicional para juzgar la calidad de una señal de video. Se basan en sus experiencias previas, y en las expectativas que tengan respecto al sistema. Los modelos del tipo SR buscan realizar un trabajo similar al del Sistema Visual Humano (SVH) (Joskowicz, y otros, 2012).

En la Universidad de las Ciencias Informática (UCI) se encuentra el centro Geoinformática y Señales Digitales (GEYSED) perteneciente a la facultad 6. Este centro cuenta con el departamento Integración de Soluciones, donde existen proyectos de desarrollo dedicados al procesamiento y transmisión de materiales audiovisuales. Como ejemplo de este tipo de proyectos se pueden mencionar: Plataforma de

[2] También llamado bit rate (BR) indica la cantidad de bits de información que se transmite en video (Castillo, 1999).

[3] Surge para intentar optimizar la utilización de la capacidad de las líneas de transmisión existentes. Se basa en la división de la información que entrega a la red el usuario emisor en paquetes del mismo tamaño que generalmente oscila entre mil y dos mil bits (Bettstetter, y otros, 1999).

Televisión Informativa PRIMICIA y Sistema de Transmisión de Canales Virtuales (STCV). El STCV está dedicado a proveer una solución tecnológica integral que permita automatizar la transmisión, administración y gestión de los canales de radio y de televisión que pudieran existir en una entidad, además de automatizar los procesos de una televisora, permitiendo una mejor calidad en las transmisiones y en la gestión de sus procesos, generando un grupo de interfaces que faciliten el trabajo de los distintos usuarios que interactúan y administran los procesos que se automatizan. Esta solución contiene cuatro subsistemas: Planificación, Transmisión, Monitorización y una Plataforma Interactiva.

El subsistema Planificación constituye la base sobre la que se realizan todas las transmisiones televisivas del sistema. Le permite al usuario de una forma sencilla e intuitiva, planificar todos los materiales que serán reproducidos dentro del entorno por cada uno de los canales. El subsistema Transmisión es el encargado de la gestión de los canales que se van a transmitir. Provee la transmisión a través de un flujo de video de los recursos multimedia según la planificación previamente establecida en el módulo de planificación. La Plataforma Interactiva es un sistema web orientado al campo de la televisión sobre el *Internet Protocol* (IP), que permite desarrollar aplicaciones interactivas de forma sencilla. Cuenta con las aplicaciones más comunes en los sistemas operativos modernos además de almacenamiento en la nube lo que le da la apariencia de un sistema operativo web.

El subsistema Monitorización se complementa con los módulos programación y transmisión, permitiendo el control constante de las transmisiones. Monitorea los flujos de video en transmisión, además de detectar los problemas que ocurran en las transmisiones planificadas en cada horario establecido. Permite visualizar en tiempo real los canales que se encuentran brindando información, además posibilita la gestión de los parámetros establecidos para cada planificación de transmisión. A través de la monitorización se puede conocer si existen paquetes perdidos en la red, la cantidad de paquetes que han sido recibidos, así como el archivo que se ha perdido en el envío de información. Sin embargo, el subsistema no cuenta con un mecanismo que le permita identificar si lo que se está transmitiendo cuenta con la calidad mínima requerida para una reproducción de video, esto trae consigo que no se pueda controlar y mejorar a su vez las transmisiones del mismo.

Ante la problemática planteada se deriva el siguiente **problema a resolver**: ¿Cómo evaluar la calidad de video en las transmisiones del Sistema de Transmisión de Canales Virtuales?

A partir del problema a resolver, se ha centrado el estudio de la solución en la evaluación de la calidad de video, lo cual constituye el **objeto de estudio**. Para resolver dicho problema, se ha identificado como

objetivo general de esta investigación, desarrollar un componente de evaluación de calidad de video utilizando la técnica Sin Referencia que contribuya a la monitorización de las trasmisiones del Sistema de Transmisión de Canales Virtuales. Enmarcado en el **campo de acción:** evaluación de la calidad de video utilizando la técnica Sin Referencia.

Luego de plantear los elementos metodológicos relacionados con la problemática que se estudia, se formulan las siguientes preguntas científicas para desglosar el problema a resolver:

1. ¿Qué métodos son utilizados para la evaluación de la calidad de video?
2. ¿Cuáles son las características de la imagen que interfieren en la calidad de video?
3. ¿Qué valores deben tomar las características de la imagen para ser evaluadas según el Sistema Visual Humano?

Para lograr el objetivo general se tendrán en cuenta las siguientes **tareas de la investigación**:

1. Establecimiento de los fundamentos teórico-metodológicos para el desarrollo de componentes de evaluación de calidad de video.
2. Caracterización de la evaluación de calidad de video en la monitorización de las transmisiones del Sistema de Transmisión de Canales Virtuales.
3. Desarrollo del componente de evaluación de calidad de video para el Sistema de Transmisión de Canales Virtuales.
4. Validación de la contribución lograda a la monitorización de las transmisiones con la introducción del componente de evaluación de calidad de video en el Sistema de Transmisión de Canales Virtuales.

Métodos científicos de la investigación

El desarrollo de la investigación está dirigido mediante métodos científicos que se aplicaron a lo largo de la misma. El método científico “se basa en la observación sistemática de la realidad, su medición, el análisis de sus propiedades y características, la elaboración de hipótesis, su interpretación y contrastación, la formulación de alternativas de acción o respuesta, para producir ciencia” (León, y otros, 2011).

Métodos teóricos

Analítico - Sintético: Se emplea para analizar los diferentes métodos y técnicas para la evaluación de la calidad de video, así como las características que inciden en la visualización del mismo, seleccionando a partir de ellas los elementos favorables para determinar la calidad de una transmisión y que a su vez se adapten a un entorno en tiempo real.

Histórico - Lógico: Permite realizar un análisis profundo sobre la evolución que han tenido los métodos y técnicas de evaluación de calidad de video, tanto subjetiva como objetiva, mostrando los experimentos realizados por grupos de investigación en el mundo y los resultados de los mismos, durante los últimos cinco años.

Métodos empíricos

Experimental: Permite validar la solución propuesta con la utilización de datos provenientes de casos reales y bajo diferentes criterios de evaluación. Se aplican pruebas estadísticas para analizar los resultados de los experimentos realizados. Se establecen indicadores que permiten realizar mediciones de los resultados.

Estructura del Documento

El documento se encuentra dividido en tres capítulos:

En el primero, Fundamentación Teórica, se realiza la elaboración del marco teórico sobre componentes de evaluación de la calidad de las transmisiones de video, abordando los conceptos fundamentales que se vinculan con el objeto de estudio de la investigación. Se realiza un análisis de la situación problemática y de las soluciones que existen en el mundo, que brinden una aproximación de lo que se pretende desarrollar. De igual manera, se definen las herramientas y el lenguaje de programación a utilizar, así como la metodología de desarrollo más recomendable. En el segundo capítulo, Presentación de la Solución Propuesta, se modela el componente, se identifican los requerimientos, se diseña la arquitectura, las clases, y finalmente, se realiza la implementación del componente para la evaluación de la calidad de las transmisiones de video. En el tercer capítulo se valida el componente desarrollado. Además, se realizan las pruebas de integración al componente, donde se muestra el resultado de las mismas, para demostrar que la solución es correcta. Finalmente, se dan las conclusiones generales y un conjunto de recomendaciones con vistas a trabajos futuros.

Capítulo 1: Fundamentación teórica

La realización de este capítulo constituye la base teórica para darle solución a la problemática existente y cumplir los objetivos planteados al inicio de la investigación. En el mismo, se abordan los elementos relacionados con la evaluación de la calidad de las transmisiones de video, explicando cada uno de ellos en cuanto a las aplicaciones prácticas que poseen, dando la posibilidad de comprender el marco de estudio donde se sustentan. Además, se expone el análisis de las soluciones desarrolladas hasta el momento.

1.1 Elementos asociados al dominio del problema

Una investigación científica origina el surgimiento de definiciones y terminologías que se desconocen, por tal motivo es necesario adentrarse en esos conceptos que sustentan el objeto de estudio que se tiene definido. Para una mejor comprensión de la situación problemática, se relacionan en lo adelante los principales conceptos que están asociados al desarrollo de la investigación.

1.1.1 Video digital

El video digital se representa como una secuencia de imágenes (fotogramas), donde además de los parámetros de resolución espacial (número de píxeles) y profundidad (número de bits), hay que considerar un número adecuado de imágenes por segundo que permitan crear la ilusión de movimiento (Agustí, y otros, 2010). El ojo tiene una propiedad llamada persistencia retiniana, de modo que cuando se ve una secuencia de imágenes que cambia rápidamente, las imágenes se superponen en la retina unas sobre otras dando la sensación de continuidad y movimiento (Montagu, 2005). También se puede arribar a otra definición de video digital, expresada como una secuencia de n fotogramas dando una sensación de movimiento a los contenidos que se encuentran en cada imagen. Cada fotograma representa la unidad básica de información del video (Lu Shi, 2011).

Fotogramas Claves

Cuando se aplica un *codec* de compresión a un video, se suele producir cierta pérdida de la información de sus fotogramas. Algunos fotogramas se almacenan completamente en el archivo comprimido, considerándose entonces fotogramas claves; mientras que el resto sólo se guardan parcialmente. En la descompresión, estos fotogramas intermedios se reconstruyen a partir de los fotogramas claves (Agustí, y otros, 2010).

Procesamiento de video

El procesamiento de video es un caso particular de procesamiento de señales, que a menudo emplea filtros de video, donde las señales de entrada y de salida son archivos de video o secuencias de video. Muchas de las técnicas de procesamiento de video se utilizan en aparatos de televisión, reproductores de video, escaladores de video y otros dispositivos. El procesamiento digital tiene su fundamento en el análisis y la interpretación de señales, para su posterior reproducción con el uso de herramientas y técnicas que lo posibiliten. Existen diversas técnicas de compresión de video, que buscan reducir el tamaño de una grabación digital para facilitar su distribución. De acuerdo al tipo de contenido, estos procesos afectan el resultado final (Barchiesi, 2008).

Imagen digital

Una imagen digital es cualquier imagen, fija o en movimiento, que se capture en un medio electrónico y que se represente como un archivo de información leído como una serie de pulsos eléctricos (Woods, y otros, 2002). Representa una matriz de m filas y n columnas, cuya intersección identifica un elemento de la imagen, y el valor correspondiente a dicho elemento indica el nivel de color en ese punto. Estos elementos de la imagen son conocidos como píxeles, por su abreviatura en inglés *picture elements* (Tovar, 2010).

Brillo: Característica de la imagen digital que da una sensación visual, en la que una zona parece emitir más o menos luz (Bezryadin, y otros, 2013).

Color: Percepción visual que se genera al interpretar las distintas distribuciones de longitudes de onda que componen la luz (Bezryadin, y otros, 2013).

Saturación: Es la intensidad de color de una imagen. Describe la intensidad (pureza), de dicho tono. Cuando el color está completamente saturado, el color se considera en la versión más pura (Bezryadin, y otros, 2013).

Contraste: Se refiere a la variación de intensidades de color o variación de niveles de escala de grises que existe en la imagen digital (Bezryadin, y otros, 2013).

1.1.2 Calidad de video

Se define calidad como propiedad o conjunto de propiedades inherentes a una cosa. Este concepto, llevado a imágenes o video, se entendería como la capacidad que una secuencia de video tiene de representar el objeto original, es decir, la exactitud o parecido entre ambos (Hormigos, 2009).

Cuando se habla de calidad, no está claro que la visibilidad del error esté relacionada con pérdida de calidad. Por ejemplo, si se considera la medida de calidad entre dos imágenes, una de las cuales es la multiplicación de los valores de luminancia de la otra imagen por un factor global, la diferencia visual entre ambas será obvia aunque el observador no aprecie tal diferencia como pérdida de calidad (Hormigos, 2009).

Evaluación de la calidad de video a través de métodos objetivos

Para poder manipular archivos de video se hace necesario conocer la forma de poder medir o evaluar la calidad de video y contenido multimedia de estos archivos, para de esta manera garantizar una óptima percepción de la información por usuarios de los diversos sistemas que gestionan este tipo de material. Para ello se hace necesario, disponer de herramientas que permitan estimar y cuantificar la calidad percibida por los usuarios en contenidos multimedia, de la manera más confiable posible.

Existen dos tipos de evaluaciones sobre la calidad de un material audiovisual, los métodos subjetivos y los métodos objetivos. Las medidas subjetivas son siempre los sistemas más confiables, ya que en dichas evaluaciones se toma directamente la percepción visual de los usuarios, teniendo una muestra considerable, la que se promedia entre un número apropiado de observaciones, obteniendo como métrica el promedio de opiniones generadas (*Mean Opinion Score*, MOS por sus siglas en inglés). Investigaciones anteriormente realizadas exponen la vía de aplicar este método, sin embargo, su realización es costosa y compleja debido a que se necesita disponer de un ambiente equipado tecnológicamente y una muestra suficientemente amplia de personas que participen en la evaluación, además de los materiales que se van a utilizar para el experimento (Joskowicz, y otros, 2012).

Además, por no obtener un resultado de evaluación en tiempo real, es que en los últimos años este amplio campo de la ciencia ha dedicado sus esfuerzos a desarrollar sistemas, algoritmos y modelos matemáticos que permitan estimar la calidad percibida por los usuarios. Un sistema ideal de estimación de calidad percibida, debería dar como resultado, una calificación idéntica a la que se obtendría en pruebas subjetivas promediando los resultados de un gran número de individuos. Esto trae consigo la utilización de

métodos objetivos que realicen una estimación automática, y puedan predecir con fiabilidad la calidad percibida.

Las medidas objetivas de calidad pueden ser clasificadas de acuerdo a la disponibilidad del video original. Éste se considera sin degradación y de calidad perfecta, y puede ser usado como referencia para comparar el video degradado. La mayoría de los métodos propuestos en la literatura asumen que la secuencia original está completamente disponible. A menudo se denomina a estas técnicas *Full Reference* (FR, por sus siglas en inglés) o de Referencia Completa (Hormigos, 2009). En estos métodos, el algoritmo tiene acceso a una versión completa y perfecta de la imagen/video original o de referencia, para ser comparada con la imagen/video procesada o degradada. Debido a la gran cantidad de recursos que se requieren, suelen ser utilizados para desarrollar algoritmos de procesamiento de imagen/video para su testeo en laboratorio, y rara vez aparecen como aplicaciones de tiempo real (Wang, 2001).

Debe tenerse en cuenta, que en la mayoría de las aplicaciones prácticas, las secuencias de referencia no serán accesibles. Por lo tanto, es muy deseable desarrollar técnicas de medida que puedan evaluar la calidad de forma ciega. Estas medidas, conocidas como Sin Referencia (SR), son muy complejas a pesar de que los observadores humanos pueden evaluar sin dificultad, y de forma efectiva y fiable la calidad de un video sin ninguna otra referencia. En estos algoritmos no se tiene acceso a la imagen/video original o de referencia. Sólo se dispone de la imagen/video procesada o degradada. Pueden ser utilizados en aplicaciones donde se necesite medir la calidad en tiempo real, puesto que consumen mucho menos recursos. La desventaja es que suelen ser menos precisos en sus predicciones de calidad, o estar únicamente diseñados para un subconjunto de imágenes/video, las que tengan un determinado defecto, o sólo un formato de imagen/video (Wang, 2001).

Existe un tercer tipo de método, en el que la señal de video original no está completamente disponible. En su lugar, ciertas características se extraen de la señal de referencia y se transmiten al sistema de medida como información adicional. Estos métodos son conocidos como Referencia Reducida (RR). En estos algoritmos, se dispone de cierta información sobre la imagen perfecta original o de referencia. Cada algoritmo define la información que necesita de la imagen/video de referencia para poder comparar con la misma, u otra información extraída de la imagen/video procesada o degradada. Se suele utilizar un canal auxiliar, llamado RR-*Channel*, para hacer llegar esta información de referencia al algoritmo de valoración de calidad (Wang, 2001).

La presente investigación centrará el enfoque de la evaluación de la calidad de las transmisiones de video en la utilización de métodos objetivos de estimación, específicamente el SR, para la obtención del resultado esperado.

1.1.3 Transmisión

Actualmente, el proceso de transmisión de las televisoras puede ser de forma tradicional o automatizada. La tradicional consiste en operar un *switch*, al cual llegan señales de video de distintas fuentes como son: grabadoras de cintas de video (VTR, siglas en inglés), señales directas del estudio o de otras televisoras. La persona encargada de realizar estas operaciones se guía por una escaleta que fue confeccionada previamente, especificando el orden en que se mostrará cada espacio y el tiempo de duración (Bustio, 2008). El avance de la informática, tanto a nivel del *hardware* como del *software* han dado la posibilidad de crear sistemas que automatizan el proceso de transmisión, siendo posible gestionar la escaleta (nombre para el guión de televisión), tener control total de las líneas de exterior y entradas de satélites, con grabación automática, titulación y gráficos, reproducción de videos desde disco duro en diferentes formatos.

1.1.4 Componente

El concepto de componentes de *software* ha cobrado fuerza debido a la importancia de su implementación sobre un producto determinado. Los mismos son una parte modular en un sistema que facilita y brinda un conjunto de funcionalidades definidas. Su utilización posibilita que sus funciones se puedan utilizar en varios sistemas sin necesidad de repetir el código; por eso, entre sus principales características, se encuentran el ser reutilizable, intercambiable, cohesivos, lo que facilita su manejo y desarrollo dentro de un entorno colaborativo. La necesidad de estandarizar los elementos de un producto trajo consigo el auge de los componentes de *software*, comenzar un proyecto desde cero es una tarea bastante difícil e innecesaria si se contara con una herramienta que facilite su realización (Montilva, y otros, 2003).

Al surgir la programación orientada a objetos se produce un cambio de paradigma muy grande en el desarrollo de *software*, donde se pasó de la descripción algorítmica de la solución, a la representación y manipulación de los objetos que forman parte del problema. Esta evolución en la programación abrió las puertas a nuevas formas de desarrollo como la basada en la reutilización de componentes mediante la creación de componentes genéricos, fáciles de integrar, distribuidos e independientes de las plataformas de desarrollo (Montilva, y otros, 2003).

1.2 Resultados obtenidos en el trabajo precedente

Al comenzar el desarrollo de la presente investigación, fue necesario analizar las condiciones en las que se encontraba el funcionamiento del sistema en cuanto a la detección de fallos en las transmisiones. Cabe mencionar que existe una investigación anterior, dirigida a la monitorización de las emisiones de audio y video en redes IP. Esta investigación surge por la necesidad de diagnosticar el estado de las emisiones de audio y video transmitidas por el STCV.

El subsistema sólo mostraba los canales en transmisión y generaba un reporte en un fichero de texto cuando estas transmisiones eran interrumpidas por algún motivo, pero no mostraba de forma visual al usuario cuándo algún canal se quedaba sin transmisión, tampoco alerta de las degradaciones existentes en la señal de video. Además, el criterio sobre la calidad de las señales emitidas se realiza de forma subjetiva. En ese momento, el subsistema Monitorización aplicaba de manera muy básica un método subjetivo de estimación, evaluado sólo por el usuario que se encontrara utilizando el sistema, que además podía visualizar en pantalla sólo un canal a la vez, lo que resultaba difícil detectar alguna irregularidad ocurrida en otro canal que no estuviese visualizando en el formulario principal.

Actualmente esas irregularidades fueron corregidas, pero la solución actual no evidencia una evaluación de la calidad de lo que se transmite, ni permite buscar una solución a tiempo durante una transmisión de baja calidad. Los parámetros que gestiona son superficiales a cualquier tipo de archivo, diagnosticando sólo el comportamiento del material a través de la red por la que es emitido. No son evaluados los metadatos ni las características del video para identificar comportamientos incorrectos o degradaciones en el mismo.

En la presente investigación se pretende lograr una solución que elimine las principales inconformidades encontradas en la solución anteriormente analizada. Para ello es necesaria la realización de un componente de evaluación de la calidad de las transmisiones de video para el subsistema Monitorización. Para lograr este cometido, se requiere el desarrollo de un método objetivo que permita evaluar la calidad de un video haciendo referencia a las características que lo componen.

1.3 Marco teórico sobre componentes de evaluación de calidad de video

No es aconsejable desarrollar una investigación sin conocer cómo marcha el mundo con respecto a lo que se pretende desarrollar, por tal motivo, es necesario conocer qué aplicaciones se han creado, dirigidas a

la producción de *software* para evaluar calidad en videos. En la búsqueda y análisis de estas soluciones, se evidencia que estos productos son desarrollados principalmente por empresas de países desarrollados, que cuentan con la tecnología y los recursos necesarios para impulsar la industria del *software* a altos niveles. La mayoría de estos productos son privativos y su utilización es muy costosa. Actualmente, el campo que se estudia se encuentra en desarrollo, por lo que se puede decir que es mucho lo que se investiga, pero que existen pocos resultados aplicables. El resultado de la búsqueda realizada se muestra a continuación, haciéndose énfasis en las principales soluciones encontradas referentes al tema de investigación que se desarrolla.

MSU Video Quality Measurement Tool (*VQMT*)

MSU *Video Quality Measurement Tool* (VQMT, por sus siglas en inglés) es una herramienta de evaluación objetiva de calidad de video, que generalmente examina estos archivos estableciendo comparaciones con el original como referencia. La función principal de este *software* es aplicar métricas de calidad objetivas de contenido multimedia digital (video o imagen) utilizando el tipo de análisis con referencia o sin referencia. Se utiliza para el análisis de calidad de *codecs* de video e imagen, para la comparación de algoritmos de procesamiento de video e imagen y para comparar la calidad de un video convertido usando distintos *codecs* (Pinson, y otros, 2002).

Entre otras de las características de VQMT, se puede mencionar que permite la visualización y puesta en práctica de numerosas métricas para evaluar calidad. Al mismo tiempo, posibilita calcular los valores de las métricas para cada marco del video y el valor promedio para la secuencia asociada a dicha imagen (Vatolin, y otros, 2010).

Los desarrolladores de dicho *software* concibieron dos versiones del producto, una propietaria que implementa varios algoritmos para procesar videos de forma objetiva como: APSNR, MS-SSIM, MSE y VQM y otra de muchas menos restricciones, pero con muy pocas funcionalidades. La versión propietaria cuenta con niveles de configuración y precisión aceptables, esta versión por ser privativa no se puede utilizar en el subsistema de monitorización. El flujo de trabajo en este subsistema no es adaptable a las configuraciones de la versión libre, debido a que el *software* cuenta con varias métricas para evaluar calidad, pero a su vez estas métricas utilizan *plugins* para realizar comparaciones en los videos y la mayoría de estos los trae inhabilitados por completo.

La arquitectura y composición de este *software* no permiten agregar funcionalidades para adaptarlo a las necesidades del sistema, además no se tiene acceso al código fuente. Esta versión libre está bajo la licencia *Creative Commons*, la cual plantea que el *software* puede ser utilizado para uso personal o de investigación, pero no con fines comerciales (Vatolin, y otros, 2010). Fue concebida con la mayoría de los *plugins* que le permiten evaluar calidad, inhabilitados, teniendo que pagar por su uso. Por estas razones no es posible utilizar esta versión en el subsistema de monitorización del STCV.

***Video Quality Experts Group* (VQEG)**

Según (VQEG, 2011) el VQEG es un organismo formado por profesionales expertos en la temática de calidad de video, provenientes de la industria, la academia y las organizaciones de estandarización. Tiene como objetivo proveer un foro apropiado para el intercambio de información y el trabajo en conjunto hacia fines relacionados con la evolución de modelos y métricas de estimación de calidad de video. VQEG ha desarrollado un enfoque sistemático para realizar la comparación del desempeño y la validación de las propuestas de modelos de estimación de calidad de video, sobre la base de contrastar los resultados de los modelos con pruebas subjetivas. Su objetivo es proporcionar información a los organismos internacionales de estandarización acerca del desempeño de diversos modelos propuestos, a los efectos de definir una métrica estándar y objetiva de calidad percibida de video digital o *Video Quality Metric* (VQM, por sus siglas en inglés).

El VQEG ha trabajado en la evaluación de modelos específicos para diferentes aplicaciones. Algunos de los proyectos del VQEG han terminado en recomendaciones aprobadas por *International Telecommunications Union* (ITU, por sus siglas en inglés). Otros se encuentran aún en proceso de evaluación o estudio. A continuación se describen los resultados obtenidos por VQEG en la aplicación de los modelos SR.

El proyecto Referencia Reducida/Sin Referencia TV (RRSN-TV) evaluó modelos del tipo RR y SR de estimación de calidad de video para aplicaciones de TV. En la fase I se evaluaron 3 proponentes, cada uno con modelos del tipo RR y SR (VQEG, 2009). Se utilizaron codificaciones en MPEG-2 y H.264/AVC. Las degradaciones incluían tanto las producidas por el proceso de codificación, como por errores en la transmisión del canal. Todos los modelos del tipo SR fueron retirados de las pruebas, y de los informes finales, debido a su bajo desempeño.

MM (MultiMedia) fue otro proyecto de este grupo de investigación, referido a aplicaciones que pueden combinar texto, gráfico, video y sonido. En la primera fase de este proyecto del VQEG, se incluyeron modelos que evalúan únicamente la calidad de video. Los resultados y el foco de la evaluación de la fase I del proyecto MM de VQEG, se centra en aplicaciones de video para dispositivos móviles o de tipo *Personal Digital Assistant* (PDA, por sus siglas en inglés) y en servicios multimedia distribuidos a través de Internet, con tasas de bits menores a 4 Mb/s, vistos en pantallas de computadoras personales con una resolución máxima de 640 × 480 píxeles.

La fase I del proyecto MM (VQEG, 2008) evaluó modelos de estimación de calidad percibida de video para aplicaciones multimedia en formatos *Video Graphics Array* (VGA, por sus siglas en inglés), 640 × 480 píxeles, *Common Intermediate Format* (CIF, por sus siglas en inglés), 352 × 288 píxeles y *Quarter Common Intermediate Format* (QCIF, por sus siglas en inglés), 176 × 144 píxeles de 25 y 30 cuadros por segundo y en modalidades FR, RR y SR. Las degradaciones introducidas incluían las propias del sistema de codificación y también errores en la transmisión. Se incluyeron secuencias codificadas en diferentes *codecs*, incluyendo H.264, H.261, H.263, MPEG4, MPEG2, Cinepak, DivX, Sorenson3 y Theora.

Se evaluaron cinco proponentes, cada uno con diversos modelos FR, RR y SR. Las empresas *Psytechnics*, *Opticom* y *Nippon Telegraph and Telephone* (NTT, por sus siglas en inglés), así como la Universidad Yonsei de Corea, presentaron modelos del tipo FR. Los resultados obtenidos en todos estos modelos, para VGA, CIF y QCIF resultaron estadísticamente mejores al PSNR. VQEG ha sugerido a ITU la estandarización de estos modelos (VQEG, 2008).

Únicamente la Universidad de Yonsei presentó modelos del tipo RR. Esta propuesta resultó estadísticamente mejor que el PSNR, siendo éste una métrica del tipo FR. VQEG ha sugerido a ITU la estandarización de estos modelos (VQEG, 2008). *Psytechnics* y *Swissqual* propusieron modelos del tipo SR. Los resultados no fueron suficientemente buenos como para que VQEG sugiriera su estandarización.

Las soluciones estudiadas no son capaces de resolver el objetivo general de esta investigación, por tal motivo es necesario desarrollar un componente de *software* que evalúe la calidad de las transmisiones de video utilizando un modelo SR.

Índice de Similitud Estructural de Video (VSSIM)

Este prototipo de *software* permite obtener el índice de calidad general para una secuencia de video. Es desarrollado en MATLAB y la elección de este lenguaje se basa en las numerosas funciones predefinidas

que incorpora. Además, este lenguaje de programación facilita realizar pruebas al instante mediante la programación en línea de comandos, sin necesidad de compilador. Los mensajes de error que este entorno de desarrollo proporciona, son de gran utilidad a la hora de corregir fallos y depurar el código realizado. En el desarrollo de la interfaz gráfica fue de gran utilidad el entorno de programación visual que ofrece MATLAB (Hormigos, 2009).

MATLAB es un *software* que utiliza bibliotecas a las cuales no se tiene acceso, lo que constituye una dificultad a la hora de integrarlo a la solución propuesta. Además, no es posible adaptar este prototipo de *software* a las características del subsistema Monitorización.

1.4 Metodología de desarrollo a utilizar

Al definir una metodología de desarrollo de *software* se pretende representar un marco de trabajo que tiene entre sus funciones: guiar, planificar, estructurar, controlar, manipular y dirigir el proceso de desarrollo de *software*; posibilitando el beneficio de trabajar mediante el uso de procedimientos, técnicas, herramientas y documentos. Las metodologías de desarrollo de *software* proveen las guías para poder conocer el camino a recorrer desde las primeras fases de la construcción del *software*, asegurando la calidad del producto final, así como también el cumplimiento en la entrega del mismo en un tiempo estipulado (Chacón, 2006).

Las metodologías se clasifican en dos tipos: ágiles y pesadas. Las metodologías ágiles tienen como principios el trabajo en equipo como arma fundamental; el avance del trabajo enmarcándose solamente en los elementos necesarios que éste exige; la constante interacción con el cliente haciéndolo parte del equipo de trabajo; la posibilidad de cambiar todo lo que debe ser cambiado de forma que se alcance la mayor fiabilidad y calidad en el producto que se desarrolla. Por su parte, las metodologías pesadas se centran en el detalle de cada proceso y tarea que se debe desarrollar, en las herramientas a utilizar, genera una documentación extensa que debe justificar cada paso de avance y además, adelanta la puesta en práctica de la solución. Se aplica fundamentalmente a proyectos grandes para realizar en igual período de tiempo y uso de recursos.

Proceso Unificado de Desarrollo (RUP)

Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés) es la metodología pesada que mayor relevancia y utilización posee en el campo del desarrollo de *software*. Define fases, principios, etapas o flujos de trabajo, disciplinas de soporte, entre otros elementos, para mejorar y organizar el trabajo desde el

comienzo (Jacobson, y otros, 1999). Su meta es asegurar la producción de *software* de alta calidad, que resuelva las necesidades de los usuarios dentro de presupuestos y tiempos establecidos. RUP es un marco de procesos altamente configurables para satisfacer necesidades específicas. Implementa las mejores prácticas del desarrollo de *software*. Es orientada a objetos y utiliza el Lenguaje Unificado de Modelado como lenguaje de representación visual. A su vez, define Quién, Cómo, Cuándo y Qué debe hacerse en el proyecto. Presentando como principales características (Chacón, 2006):

Dirigido por casos de uso: Utiliza los casos de uso para el desarrollo de las disciplinas con los artefactos, roles y actividades necesarias. Además, éstos son la base para la implementación de las fases y disciplinas de RUP.

Iterativo e Incremental: Plantea la implementación del proyecto a realizar en iteraciones, por lo que se pueden definir objetivos por cumplir en cada iteración y así completar el proyecto iteración por iteración. Ejecuta una evaluación satisfactoria, permitiendo que el proyecto se mueva a la próxima fase y realiza una revisión del ciclo de vida al finalizar la misma.

Centrado en la Arquitectura: Define la estructura de las partes más relevantes de un sistema y una arquitectura ejecutable construida como un prototipo evolutivo. RUP es uno de los procesos más generales de los existentes en la actualidad, debido a que está pensado para adaptarse a cualquier proyecto.

Aunque el desarrollo de la solución se pretende que no sea demasiado costosa en cuanto a fuerza de trabajo y artefactos a generar, se selecciona RUP como metodología de desarrollo cumpliendo con las políticas del proyecto para el cual se está realizando esta investigación. De forma tal, que la construcción de la solución respete los estándares y políticas establecidas en el proyecto sobre el desarrollo de *software*. Además, los artefactos generados deben estar en correspondencia con lo definido en el expediente de proyecto, destacando que el principal factor para su selección se debe a los beneficios que aporta al desarrollo de la futura solución.

(Larman, 2003) explica "(...) *Sin embargo, es importante saber que en el Proceso Unificado (UP, por sus siglas en inglés) todas las actividades y artefactos (modelos, diagramas, documentos...) son opcionales. El conjunto de artefactos posibles del UP debería entenderse como un conjunto de medicinas en una farmacia. Exactamente igual que uno no toma medicinas indiscriminadamente, sino que las elige según la dolencia, en un proyecto UP, un equipo debería seleccionar un pequeño subconjunto de artefactos que*

sirvan para tratar sus problemas y necesidades particulares. En general, centrarse en un pequeño conjunto de artefactos que demuestran tener un gran valor práctico. (...)".

Este planteamiento defiende el concepto de adaptar los artefactos generados por la metodología seleccionada según las necesidades del equipo de desarrollo y en particular con el tipo de solución que se realiza. Esto facilitará el empleo de RUP en la presente investigación.

1.5 Tecnologías a utilizar en el desarrollo de la solución

Inmersos en la batalla por la utilización del *software* libre, en Cuba se impulsa el desarrollo de la informática bajo esta premisa, con el fin de poder proveer al país de un soporte tecnológico basado en código abierto, que el *software* propietario obstaculiza. Por lo anteriormente planteado, se hace necesario definir las herramientas a utilizar en el desarrollo de la investigación que se correspondan con la libertad tecnológica que se necesita y que cumplan con las definidas por el proyecto STCV, garantizando compatibilidad y entendimiento de lo que se desarrolla.

1.5.1 Framework de desarrollo

La utilización de un *framework* o marco de trabajo es fundamental para el desarrollo de *software*, ya que representa una estructura conceptual y tecnológica que provee artefactos o módulos de *software*, sirviendo de base para la organización y desarrollo de la solución. El mismo, puede incluir lenguaje interpretado, bibliotecas, soporte a programas y otras herramientas para unir los componentes que sustentan un proyecto (Medina, 2014).

Qt como *framework* de desarrollo

Tiene la característica de ser multiplataforma, se emplea en mayor medida para el desarrollo de herramientas. Fue construido con el uso de C++ y tiene soporte para varios lenguajes de programación. Con el fin de lograr el desarrollo de una aplicación que cumpla con las necesidades del proyecto, se utiliza Qt como *framework* de desarrollo. Una característica muy importante que maneja Qt es el paradigma señales y *slots*, a través del cual los objetos de una aplicación se comunican. Además de brindar facilidades de uso, este *framework* posee una documentación extensa que posibilita el entendimiento, aprendizaje y desarrollo de su utilización para los usuarios (Ramiro, 2011).

La selección de Qt como *framework* está dada por las características funcionales que presenta, además de que facilita el uso con el lenguaje C++ y que responde a las tecnologías definidas para la construcción de los productos del proyecto.

1.5.2 Lenguaje de programación

Un lenguaje de programación es el idioma que utilizan las computadoras para construir la interacción con los usuarios, de esta forma se pueden emplear para crear programas, algoritmos, que solucionen problemas reales. En su estructura se definen símbolos, reglas sintácticas y semánticas, que justifican su comportamiento. Tiene procesos asociados como la depuración, la compilación, la escritura, que unidos en su conjunto forman el concepto de programación. De acuerdo al nivel de abstracción, se habla de lenguaje de máquina, lenguaje de bajo, medio y alto nivel.

C++ como lenguaje de programación

C++ es un lenguaje de programación que soporta la programación orientada a objetos y la programación estructurada, se le considera un lenguaje híbrido. Actualmente es uno de los lenguajes más potentes y esto se debe en gran medida a que es un lenguaje compilado. Otra de sus particularidades es que brinda la posibilidad de redefinir los operadores y de crear nuevos tipos de datos. Permite la reutilización de código en una forma lógica y productiva (Katrib, 1997).

Para el desarrollo del componente de evaluación de calidad de video se necesita un lenguaje que sea rápido, eficiente, que brinde funciones al equipo de desarrollo de forma tal, que la construcción del *software* sea fluida y sin trabas. El hecho de ser compilado y no interpretado, le da una condición de rapidez y liderazgo sobre otros, posibilitando el procesamiento de videos con la utilización de algoritmos que necesiten un alto rendimiento. Con su selección se garantiza la adaptabilidad al sistema para el que se está realizando la presente investigación, debido a que éste es implementado sobre este mismo lenguaje. Además, la biblioteca que se utilizará para el procesamiento de los fotogramas, *OpenCV*, emplea C++ como lenguaje de programación.

1.5.3 Entorno Integrado de Desarrollo

Teniendo seleccionados en los sub-epígrafes anteriores el *framework* de desarrollo y el lenguaje de programación, se hace más sencillo escoger cuál será el IDE a utilizar en el desenlace de la presente investigación. Se hace necesario seleccionar un IDE que tenga soporte para el lenguaje C++, buen completamiento de código, integración con bibliotecas de Qt, consumo de memoria reducido y abundante

documentación, estas características llevan al estudio de *QtCreator*. Éste contiene todas las herramientas necesarias para crear cualquier tipo de aplicaciones desde un mismo entorno de desarrollo. Desde el mismo IDE se puede acceder a toda la documentación del *framework*, incluso desde el mismo código se puede consultar lo que realiza una determinada clase o función.

QtCreator es un IDE multiplataforma que posee un avanzado editor de código C++, lenguaje que utiliza de forma nativa. Al utilizar el lenguaje de programación C++, se hace uso de la programación orientada a objetos. *QtCreator* posee herramientas como: *QtAssistant*, que permite interactuar con la documentación de Qt, donde se muestran todos los elementos que se deben dominar para el trabajo con el IDE; *QtLinguist*, permite crear aplicaciones con soporte para varios idiomas (Ramiro, 2011).

1.5.4 Biblioteca a utilizar

La utilización de bibliotecas adicionales brinda un marco de trabajo de alto nivel para el desarrollo de aplicaciones. Posibilita el procesamiento y visualización de imágenes, estructura de datos y el análisis de dichas estructuras. Cuando se utiliza un IDE como entorno de desarrollo, en este caso *QtCreator*, el uso de bibliotecas incorpora funciones que viabilizan el procesamiento de imágenes para la implementación del sistema.

Biblioteca de Código Abierto de Visión por Computadora

Biblioteca de Código Abierto de Visión por Computadora (OpenCV, por sus siglas en inglés) es una biblioteca de código abierto que proporciona un gran número de funciones para el procesamiento de imágenes. Permite a los programadores crear aplicaciones poderosas en el dominio de la visión digital. Ofrece muchos tipos de datos de alto nivel como árboles, gráficos, matrices, entre otros. Incluye funcionalidades como: realizar operaciones básicas en matrices, procesado de imágenes, análisis de movimiento, segmentación y reconocimiento de objetos, entre otras.

Esta biblioteca implementa gran variedad de herramientas para la interpretación de la imagen. Además de incluir operaciones primitivas como estadísticas de la imagen y filtrado, las cuales son utilizadas en la implementación. Los algoritmos están basados en estructuras de datos muy flexibles. Brinda funciones que posibilitan convertir una imagen en diferentes formatos de color (OpenCV, 2003).

Características de OpenCV como la incorporación de estructuras de la imagen para facilitar su manipulación digital, realizar operaciones básicas entre matrices, procesado de imágenes y conversión del formato de color, influyeron significativamente en su elección para el desarrollo de la solución.

1.5.5 Lenguaje Unificado de Modelado

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es un lenguaje visual para especificar, construir y documentar esquemas de los sistemas de *software*. El mismo contiene diagramas que permiten la modelación de los diferentes componentes que integran el sistema. Es utilizado por metodologías de desarrollo, presentando una estrecha relación con RUP, que es la metodología a utilizar en la presente investigación (Orallo, 2003). El uso de UML indica lo que supuestamente hará el sistema, pero no cómo lo hará; incluye estereotipos como mecanismos de extensibilidad y además, puede describir cualquier tipo de sistemas en términos de diagramas orientados a objetos.

El desarrollo de sistemas con este lenguaje permite divisar con mayor facilidad las dificultades y dependencias implícitas en la solución informática. Incluye actividades específicas que sirven de guía para saber cómo deben ser las tareas desarrolladas.

1.5.6 Herramienta CASE para el modelado

Las herramientas CASE (del inglés *Computer Aided Software Engineering*) representan un conjunto de programas y ayudas que dan soporte y asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todo el ciclo de desarrollo del sistema (Alfaro, 1999). Este tipo de herramientas facilitan el proceso de desarrollo de *software* con el aumento de la productividad y la reducción del tiempo que se utiliza. Utilizándolas se puede diseñar, implementar a partir del diseño realizado, compilar automáticamente, detectar errores y brindarle seguridad al equipo de trabajo sobre el avance de la solución.

***Visual Paradigm* para UML**

Es una herramienta CASE multiplataforma que soporta múltiples usuarios trabajando sobre el mismo proyecto. Brinda soporte para el modelado y visualización de componentes necesarios durante todo el ciclo de vida del *software*. Constituye una plataforma para el desarrollo de sistemas con bajos costos y buena calidad. Permite representar todo tipo de diagramas UML. Se considera muy completa y fácil de usar, con soporte multiplataforma y excelentes facilidades de interoperabilidad con otras aplicaciones. Posibilita la captura de requerimientos, análisis, diseño e implementación para una aplicación en desarrollo.

Esta herramienta tiene la capacidad de generar código y realizar ingeniería inversa con diferentes lenguajes de programación orientados a objetos. Posibilita la integración con diferentes entornos de

desarrollo integrados y otras herramientas CASE, brindando facilidades para la exportación e importación de componentes. Esta herramienta de modelado ha ganado gran potencial, ya que es utilizada para el desarrollo de sistemas en plataformas libres y a pesar de su licencia comercial, su versión *Community* posee licencia gratuita.

Entre las ventajas que ofrece, están (Sierra, 2006):

- Generación de código: Modelo a código, diagrama a código, para diferentes lenguajes, entre ellos C++, Java y exportación como HTML.
- Interoperabilidad entre diagramas: Permite a partir de un diagrama, obtener otro que guarde relación con el mismo.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Generación de documentación: Brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Lo anteriormente expuesto, demuestra que *Visual Paradigm* es la mejor alternativa de herramienta CASE para un desarrollo exitoso del componente que se implementa, ya que agilizará el proceso de desarrollo y generará los estereotipos necesarios con la estructura y relaciones deseada.

1.6 Conclusiones parciales

En el desarrollo de este capítulo se abordaron aspectos significativos que sirvieron para entender el entorno científico donde se encuentra enmarcada la presente investigación y que además, demuestran la necesidad de su realización. El estudio del estado del arte sobre soluciones similares evidencia la inexistencia de aplicaciones, componentes, sistemas que solucionen el problema científico expuesto a lo largo de la investigación. Las condiciones de trabajo sobre la que está respaldada la presente investigación conllevan al uso de RUP como metodología de desarrollo. Al tener en cuenta las políticas del departamento Integración de Soluciones y las particularidades de la solución propuesta, se decide utilizar el *framework* Qt, el IDE *QtCreator*, el lenguaje de programación C++, la biblioteca *OpenCV*, además de *Visual Paradigm* como herramienta CASE, utilizando UML como lenguaje de modelado.

Capítulo 2: Presentación de la solución propuesta

En el presente capítulo se realiza una descripción de la solución propuesta, teniendo como guía la metodología de desarrollo de *software* seleccionada en el capítulo anterior. Con el fin de lograr un mayor entendimiento del entorno del problema a resolver, se presenta un modelo de dominio que describe detalladamente cada una de sus entidades identificadas. Se realizará el levantamiento de los requerimientos que debe cumplir la solución propuesta, así como la elaboración y especificación de los casos de uso correspondientes.

2.1 Modelo de dominio

El modelo de dominio es la representación visual de los conceptos u objetos del mundo real en un dominio de interés. En él se representan los conceptos del dominio que resultan importantes, sus características y las relaciones entre dichos conceptos (Giraldo, 2009). Se muestra como un conjunto de diagramas de clases en los que no se define ninguna operación. Este artefacto es definido por RUP dentro de la disciplina de modelado del negocio (Larman, 2003). En la presente investigación el modelado del negocio no se realiza en su totalidad, ya que al no existir un negocio real, se imposibilita precisar la estructura de los procesos de negocio. El STCV presenta un desarrollo adaptable a cualquier entidad que requiera de los procesos que él mismo implementa, es por ello además, que no se considera necesario generar todos los artefactos que propone el modelado del negocio y se parte del estudio del dominio en que se encuentra el problema a resolver.

2.1.1 Descripción general del modelo de dominio

El monitor u operador de Monitorización es el encargado de controlar las operaciones, como insertar, eliminar y monitorizar canales de televisión que se realicen dentro del subsistema Monitorización, y garantizar que las transmisiones se lleven a cabo correctamente. El subsistema permite la monitorización constante de las transmisiones que se realizan. Es posible visualizar en tiempo real cada uno de los canales, en caso que lo requiera se puede obtener mayor detalle al seleccionar uno de ellos. El sistema es capaz de actualizar los flujos de video de cada canal, así como los datos asociados a éste, además, emite una alarma visual de error en caso que se produzca alguna falla en las transmisiones.

2.1.2 Diagrama de clases del modelo de domino

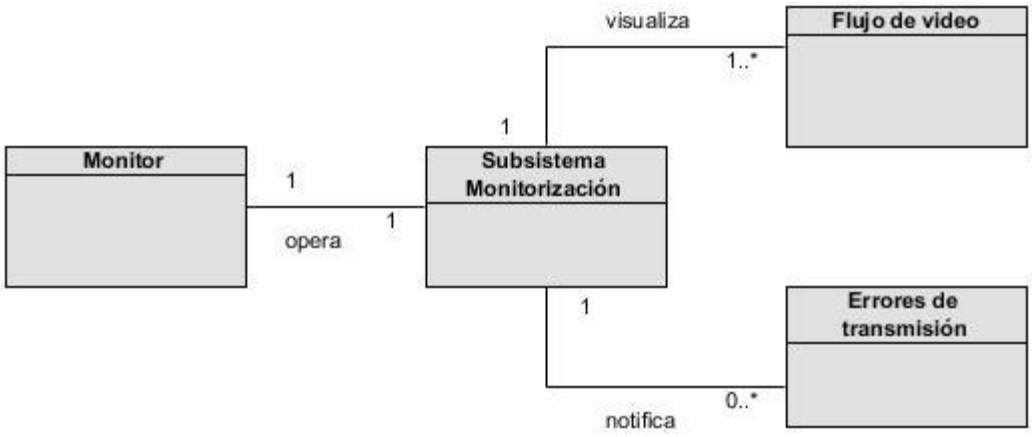

Fig 1: Modelo de dominio.

2.1.3 Descripción de las clases

Monitor u Operador de monitorización: Persona encargada de controlar las operaciones que se realicen dentro del subsistema Monitorización, y garantizar una adecuada transmisión, con la vigilancia constante de lo que ocurre en cada uno de los canales.

Subsistema Monitorización: Subsistema que se encarga de la visualización de los flujos que están siendo transmitidos por el subsistema Transmisor o cualquier flujo externo. Tiene implementado a su vez la notificación de errores que puedan ocurrir durante el proceso de transmisión, de forma que el operador de la monitorización tenga el control de todo lo ocurrido en las transmisiones y pueda tomar medidas con el fin de mejorar la calidad del servicio que se brinda.

Errores de transmisión: Notificaciones enviadas por el subsistema Monitorización, que permiten conocer los problemas ocurridos durante la transmisión de un material.

Flujo de video: Transmisión multimedia que se recibe luego de la emisión por parte del subsistema Transmisión o cualquier flujo externo equipado para este fin.

2.2 Identificación de los requerimientos de software

Para el desarrollo de cualquier proyecto, tanto informático como social, es necesario tener como punto de partida, cuáles son las capacidades y características que el mismo requiere para su realización fructífera. De otra manera, resulta ineficiente comenzar una tarea sin saber qué se quiere obtener de ella y bajo qué condiciones. En la informática se emplea el término requerimiento para expresar lo que debe hacer el

sistema, que se convierte en definir los requerimientos funcionales, además de especificar los requerimientos no funcionales que son propiedades o cualidades que el producto debe cumplir.

2.2.1 Requerimientos funcionales del componente de evaluación de calidad

Los requerimientos funcionales (RF) de un sistema describen lo que el sistema debe hacer. Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (Sommerville, 2000).

RF 1: Capturar flujo de video.

Descripción: Cuando el sistema inicia una transmisión y el monitor acude a la opción evaluar calidad, el componente de evaluación de la calidad debe poder capturar este flujo de video para procesarlo y emitir la evaluación de su calidad.

Entradas: Dirección origen del flujo de video que se desea capturar.

Salidas: Lista con los fotogramas capturados.

RF 2: Determinar fotogramas claves.

Descripción: Luego de capturar una parte del flujo de video, es necesario que el componente obtenga un fotograma por cada escena para ser procesado, ganando en tiempo de cómputo y con ello, conocer la evaluación en tiempo real.

Entradas: Lista con los fotogramas capturados.

Salidas: Lista con los fotogramas claves.

RF 3: Evaluar brillo del fotograma.

Descripción: El componente debe determinar los niveles de brillo de cada fotograma clave, para luego determinar la evaluación de la imagen teniendo en cuenta los niveles de brillo obtenidos.

Entradas: Lista con los fotogramas claves.

Salidas: Lista de evaluaciones de cada fotograma clave procesado.

RF 4: Evaluar color del fotograma.

Descripción: El componente debe determinar los niveles de color de cada fotograma clave, para luego determinar la evaluación de la imagen teniendo en cuenta los niveles de color obtenidos.

Entradas: Lista con los fotogramas claves.

Salidas: Lista de evaluaciones de cada fotograma clave procesado.

RF 5: Evaluar saturación del fotograma.

Descripción: El componente debe determinar los niveles de saturación de cada fotograma clave, para luego determinar la evaluación de la imagen teniendo en cuenta los niveles de saturación obtenidos.

Entradas: Lista con los fotogramas claves.

Salidas: Lista de evaluaciones de cada fotograma clave procesado.

RF 6: Evaluar contraste del fotograma.

Descripción: El componente debe determinar los niveles de contraste de cada fotograma clave, para luego determinar la evaluación de la imagen teniendo en cuenta los niveles de contraste obtenidos.

Entradas: Lista con los fotogramas claves.

Salidas: Lista de evaluaciones de cada fotograma clave procesado.

RF 7: Notificar resultado de la evaluación.

Descripción: Como etapa final de la evaluación de la calidad de una transmisión de video digital, se realiza la notificación de la evaluación de cada fotograma procesado.

Entradas: Lista de Lista de evaluaciones de cada fotograma clave procesado.

Salidas: Lista de evaluaciones de cada fotograma clave procesado.

2.2.2 Requerimientos no funcionales

Los requerimientos no funcionales (RNF), como su nombre lo indica, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los RNF a menudo se aplican al sistema en su totalidad.

Normalmente, apenas se aplican a características o servicios individuales del sistema (Sommerville, 2000).

RNF 1: Usabilidad

- Tipo de usuario final

Descripción del requerimiento: El monitor u operador de la monitorización debe ser técnico medio en informática o graduado universitario con una especialidad vinculada al procesamiento de materiales audiovisuales, para poder entender el resultado de evaluación de un archivo de video digital y determinar una solución en caso necesario.

- Tipo de Aplicación Informática

Descripción del requerimiento: Por las funciones que debe brindar el componente desarrollado, debe ser una aplicación de escritorio.

- Finalidad

Descripción del requerimiento: La aplicación tiene como objetivo la evaluación en tiempo real de la calidad de un video digital durante una transmisión.

- Ambiente

Descripción del requerimiento: Los requerimientos de ambiente que se describen a continuación se deben a la necesidad de procesamiento rápido que tiene el componente, garantizando una ejecución eficiente de los procesos que tiene implementados.

Requerimientos de *Software*:

- Sistema Operativo: Ubuntu 12.04 ó superior
- *Framework* Qt v5.3
- Biblioteca *OpenCV* v2.4.9

Requerimientos de *Hardware*:

- 2 GB de memoria RAM
- Procesador Intel *Dual Core* 2.2 GHz
- Tarjeta de red 100 mbps

RNF 2: Confiabilidad

- Requerimiento de Fiabilidad

Descripción del requerimiento: El componente debe ser capaz de funcionar todo el tiempo. En caso de detenerse la transmisión, el componente evaluará dicha transmisión hasta el momento que estuvo disponible.

RNF 3: Eficiencia

- Requerimiento de Eficiencia

Descripción del requerimiento: La evaluación se realizará en tiempo real, siempre teniendo en cuenta que el procesamiento de imágenes tiende a ser lento, dependiendo de los recursos con los que se disponga, y el tiempo de respuesta tendrá un desfasaje.

RNF 4: Restricciones de diseño e implementación

- Requerimiento de diseño e implementación

Descripción del requerimiento: Los requerimientos que a continuación se exponen garantizarán el correcto funcionamiento del componente y su escalabilidad en cuanto al diseño de implementación.

  - El componente será implementado siguiendo el paradigma de la Programación Orientada a Objetos.
  - Para la modelación del componente se utilizará como lenguaje de modelado UML 2.1.
  - Se requiere el uso de los estilos arquitectónicos Llamada y Retorno, y Tuberías y Filtros.
  - Será implementado en el lenguaje de programación C++, utilizando el *framework* Qt v5.3, el IDE *QtCreator* v3.0 y las bibliotecas *OpenCV* v2.4.9.

- Requerimiento de Integración

Descripción del requerimiento: El componente debe permitir integrarse al STCV mediante el uso de una interfaz de comunicación que sirve para recibir y enviar información en ambas direcciones.

2.3 Descripción del componente de evaluación de calidad de video

Luego de definir los requerimientos con los que cuenta el componente de evaluación de calidad de video digital, se presenta, haciendo uso del lenguaje de modelado UML, el diagrama de casos de uso del componente. En este diagrama, cada uno de los requerimientos abarca una serie de acciones del

componente, las cuales se llevan a cabo por medio de un actor. Un actor es un elemento que posee un comportamiento, incluyendo el propio sistema cuando solicita los servicios de otros sistemas (Larman, 2003).

2.3.1 Actor del componente de evaluación de calidad de video

Para el desarrollo del componente sólo se evidencia la incidencia de un actor, que tiene la función de monitor u operador de la monitorización. Este actor es la persona encargada de supervisar el subsistema Monitorización en su conjunto, además de darle al componente desarrollado la función de evaluar la calidad del video digital que se está transmitiendo. Durante el procesamiento, se notifican las evaluaciones obtenidas de la transmisión, ayudando a tomar una decisión al monitor del sistema. Este actor es el encargado además de detener el procesamiento y con ello la evaluación.

Diagrama de Casos de Usos del sistema

El caso de uso describe la secuencia de eventos de un actor que utiliza un sistema para completar un proceso (Jacobson, y otros, 1999). Los casos de uso son historias o casos de utilización de un sistema; no son exactamente los requerimientos ni las especificaciones funcionales, sino que ejemplifican e incluyen tácticamente en los requerimientos en las historias que narran (Larman, 2003). Son mecanismos ampliamente utilizados para descubrir y registrar los requerimientos.

En este epígrafe se muestra la relación del actor con el caso de uso del sistema (CUS) identificado durante la realización del modelo del sistema.

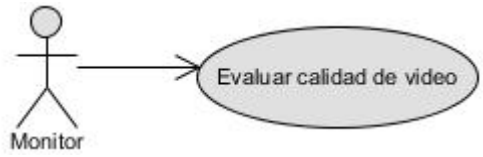

Fig 2: Diagrama de Casos de Uso del componente.

2.3.2 Descripción textual de CUS

CU 1. Evaluar calidad de video

Tabla 1: Descripción del CU Evaluar calidad de video.

Objetivo	Evaluar la calidad de un video digital durante su transmisión por un canal televisivo.
Actores	Monitor: (Inicia) Ejecuta la opción Evaluar calidad de video en el subsistema Monitorización.
Resumen	El CU inicia luego que el monitor ejecuta en el subsistema Monitorización la opción Evaluar

	calidad de video. Seguidamente el sistema captura el flujo de video, detecta los fotogramas claves y los evalúa. El CU termina cuando el monitor ejecuta la opción Detener evaluación, en el subsistema Monitorización.
Complejidad	Alta.
Prioridad	Crítico.
Precondiciones	El subsistema Monitorización tiene una transmisión visualizándose.
Postcondiciones	Notifica al monitor mediante el subsistema Monitorización el resultado de la evaluación de cada fotograma procesado.

Flujo de eventos

Flujo básico: Evaluar calidad de video

	Actor	**Sistema**
1.	Accede a la opción Evaluar calidad de video.	
2.		Envía una señal al componente de evaluación de calidad de video, a través de una interfaz de comunicación con la dirección de la transmisión, para su captura y evaluación.
3.		Captura los fotogramas que componen la secuencia de video que se está analizando, almacenándolos en una lista.
4.		Determina los fotogramas claves y los almacena en otra lista para procesarlos.
5.		Realiza el procesamiento siguiente sobre los fotogramas claves almacenados: Evaluar brillo, color, saturación, contraste.
6.		Notifica al monitor mediante el subsistema Monitorización el resultado de la evaluación de la calidad de video digital.
7.		Termina el CU.

Flujos alternos

1a Evento Detener evaluación

	Actor	**Sistema**
1.	Detener el procesamiento para la evaluación de la calidad de video.	
2.		Termina el CU.

Relaciones	**CU incluidos**	No existen.
	CU extendidos	No existen.

2.4 Modelo de análisis

El modelo de análisis es una representación abstracta del sistema que se modela, en el que se captura la lógica esencial del mismo sin entrar en detalles. En él se analizan los requerimientos, se refinan y estructuran, utilizando el lenguaje de los desarrolladores para describir cómo debería diseñarse e implementarse (Jacobson, y otros, 1999). Sin embargo, este procedimiento no es obligatorio, la metodología RUP es altamente configurable y hay artefactos que pueden obviarse durante el desarrollo de *software*. Las clases del análisis luego evolucionan hacia otras clases más detalladas en el modelo del diseño.

Debido a esto, se decide hacer una transición del modelo de casos de uso al modelo de diseño, sin necesidad de realizar el modelo de análisis, teniendo en cuenta que los requerimientos son conocidos, que se tiene conocimiento sobre los procesos que se desarrollan y que, el lenguaje de programación, el *framework* y en general las tecnologías sobre las que se estará desarrollando el sistema, son elementos dominados.

2.5 Arquitectura de software

La arquitectura de *software* es una estructura de alto nivel de un *software* y sus propiedades globales, que parte del diseño de *software* y define el nivel de diseño basado en la estructura y propiedades globales del sistema que se desarrolla. Incluye componentes, propiedades observables de dichos componentes y las relaciones que se establecen entre ellos (Bass, y otros, 2003). Identificar una arquitectura erróneamente puede llevar a problemas incontables en el proceso de construcción del sistema.

El tópico más urgente y exitoso en arquitectura de *software* en los últimos años es el de los patrones arquitectónicos. Inmediatamente después, en una relación a veces de complementariedad, otras de oposición, se encuentra la sistematización de los llamados estilos arquitectónicos. Los estilos favorecen un tratamiento estructural que concierne más bien a la teoría, la investigación académica y la arquitectura en el nivel de abstracción más elevado, mientras que los patrones se ocupan de cuestiones que están más cerca del diseño, la práctica, la implementación, el proceso, el refinamiento, el código (Reynoso, y otros, 2004).

El componente que se propone como solución en la presente investigación, necesita ser integrado al STCV para resolver el problema que dio origen a su implementación, por lo que debe describir una

arquitectura adecuada que permita la ejecución de sus funciones integradas. A continuación se detallan los elementos que sustentarán la arquitectura seleccionada.

Estilos y patrones arquitectónicos

Un estilo describe una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas (Kazman, 2001). Fundamentan la relación de cada funcionalidad de manera estructurada y jerárquica, son artefactos de ingeniería importantes porque definen clases de diseño junto con las propiedades conocidas asociadas a ellos. Ofrecen evidencia basada en la experiencia sobre la forma en que se ha utilizado históricamente cada clase, junto con razonamiento cualitativo para explicar por qué cada clase tiene esas propiedades específicas (Reynoso, y otros, 2004).

Para el desarrollo del componente de evaluación de la calidad de video digital en transmisiones en vivo, se emplea el estilo Llamada y Retorno. El componente que se desarrolla necesita recibir una llamada del subsistema Monitorización para ejecutar la evaluación de la calidad, donde, mediante una interfaz de comunicación, obtiene la dirección de la transmisión que está disponible en ese momento, para sobre ella, aplicar las funciones implementadas en el componente resultante de la presente investigación. Luego de realizar todo el procesamiento para la evaluación, el componente retorna al subsistema una respuesta como resultado de la llamada realizada por el mismo, cumpliéndose así con los principios que describen al estilo Llamada y retorno.

Para una mejor estructuración del componente, se hace necesario utilizar una Arquitectura de Flujo de Datos. Las arquitecturas de flujo de datos no se basan en un contador de programa, en tanto la posibilidad de ejecución de las instrucciones solamente viene determinada por la disponibilidad de los argumentos de entrada de las instrucciones (Reynoso, y otros, 2004).

El uso de esta arquitectura provee el estilo arquitectónico: Tuberías y filtros. Una tubería es una popular arquitectura que conecta al componente computacional a través de conectores, ejecutándolo como un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas (Shaw, y otros, 1996). Los procesos que implementa el componente se derivan de un flujo de datos que es filtrado hacia las diferentes funciones que contiene el componente. Específicamente, el flujo de video capturado por el componente, es preparado fotograma a fotograma, enviando cada uno de ellos a varios filtros como: determinar fotogramas claves, evaluar brillo, evaluar color, entre otros; como

parte del procesamiento para la evaluación de la calidad de video, cumpliendo así con las características del estilo arquitectónico Tuberías y Filtros.

2.5.1 Patrones de diseño de la arquitectura

Los patrones de diseño brindan una solución a problemas comunes en el desarrollo de *software* (Reynoso, y otros, 2004). Entre los patrones de diseño más utilizados se encuentran los patrones *General Responsibility Assignment Software Patterns* (por sus siglas en inglés, GRASP) y los patrones *Gang of Four* (por sus siglas en inglés, GoF). Durante el desarrollo de la investigación fueron utilizados los siguientes patrones de diseño:

Patrones GRASP

Su nombre se eligió para indicar la importancia de diseñar eficazmente el *software*. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades, expresados en forma de patrones. Los patrones GRASP que serán utilizados son (Larman, 2003):

- Experto: Está diseñado para que la responsabilidad de realizar una labor sea de la clase que tiene o puede tener los atributos involucrados, mantiene la responsabilidad dentro de las mismas clases que contienen los diversos atributos y las diversas propiedades, logrando que exista una alta cohesión y un mejor desempeño. En el componente para la evaluación de la calidad de video digital de las transmisiones del STCV se evidencia el empleo de este patrón en la clase *hiloDeterminarCalidadVideo*, la que contiene toda la información necesaria para realizar las funcionalidades que implementa; obteniéndose un diseño con mayor cohesión.
- Creador: Es el patrón que ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos o clases, basándose en reconocer si dicho creador tiene la información suficiente y necesaria para realizar dicha actividad, que use alguna instancia del objeto que cumple el rol de creado y que almacene o maneje varias instancias. En el componente se evidencia el empleo de este patrón en la clase *hiloCapturarStreaming*, donde se instancia un objeto de la clase *VideoCapture* de la biblioteca *OpenCV*, para capturar el flujo que se transmite y desencadenar el proceso de evaluación de calidad de video.
- Controlador: sirve de puente de comunicación entre la interfaz *IPlugin* y el componente de evaluación de la calidad de video, de forma que mediante esta interfaz, la clase *Principal* recibe los

datos del STCV, para comenzar el procesamiento de la evaluación. A su vez, mediante este puente de comunicación, el componente devuelve al sistema la evaluación.

- Alta Cohesión: Dicho patrón tiene como objetivo asignarle a cada clase sólo la responsabilidad necesaria para su correcto funcionamiento, no asignar más eventos de los que ya contiene y a la vez permitir que cada clase pueda colaborar con otras para realizar diferentes operaciones tales como: instanciar objetos y acceder a las propiedades. Este patrón se evidencia en cada una de las clases del componente.
- Bajo Acoplamiento: Permite establecer las conexiones entre las clases sólo cuando es necesario, manteniendo un bajo acoplamiento en las diversas responsabilidades y que cuando se realice alguna modificación a un objeto específico, se tenga la mínima repercusión en el resto de los objetos que puedan contener al modificado y permitiendo disminuir la dependencia entre éstos. Este patrón se evidencia en cada una de las clases del componente.

Patrones GoF

Estos patrones de diseño se dividen en tres categorías (Gamma, y otros, 2003):

- Patrones de creación: Abstraen el proceso de creación de objetos y ayudan a crear sistemas independientes de cómo los objetos son creados, compuestos y representados.
- Patrones estructurales: Definen cómo clases y objetos se combinan para formar estructuras más complejas. Son patrones basados en la herencia.
- Patrones de comportamiento: Están relacionados con la asignación de responsabilidades entre clases. Enfatizan la colaboración entre objetos.

El patrón de diseño Observador (en inglés *Observer*) se encuentra en la categoría de patrón de comportamiento. Permite distribuir el código realizado en varias direcciones sin tener que modificarlo y establece una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos. Su intención fundamental consiste en proporcionar a los componentes una forma flexible de enviar mensajes de difusión a los receptores interesados. Se evidencia el patrón Observador, en el envío y recepción de los datos entre el componente y el subsistema Monitorización a través de la clase *IPlugin*. A su vez, se usa este patrón en el envío de mensajes entre las clases del componente, implementadas por el mecanismo *signals-slot*.

2.6 Diagrama de clases del diseño

Un diagrama de clases del diseño tiene como función representar, de manera simple y de fácil comprensión, los atributos y métodos principales de cada una de las clases que componen el sistema, sus relaciones y responsabilidades (Jacobson, y otros, 1999).

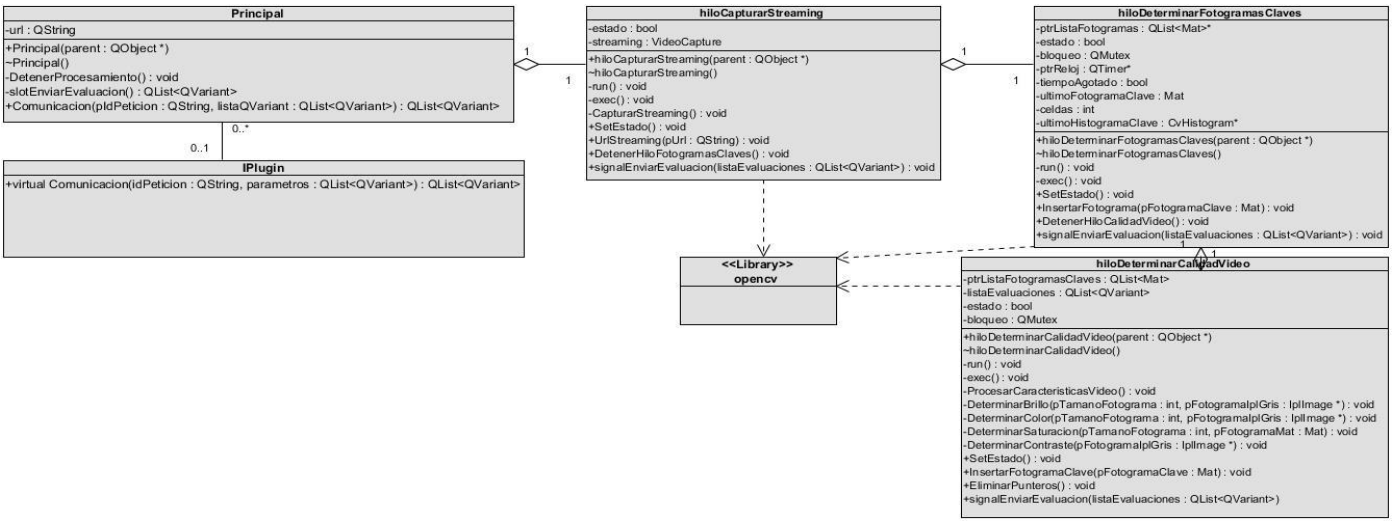

Fig 3: Diagrama de clases del diseño del componente.

En la figura anteriormente representada, se muestra el diagrama de clases del diseño para el componente de evaluación de calidad de video. En este diagrama interviene la clase *IPlugin* que actúa como puente de comunicación entre el subsistema Monitorización y la clase *Principal*. La clase *Principal* delega la funcionalidad de capturar el flujo de video en la clase *hiloCapturarStreaming*, y ésta a su vez conecta con la clase *hiloDeterminarFotogramasClaves*, para añadir a la lista *ptrListaFotogramas* cada fotograma que se capture del flujo, y que sea procesado para determinar si es fotograma clave. Luego, la clase *hiloDeterminarFotogramasClaves* conecta con la clase *hiloDeterminarCalidadVideo* para añadir a la lista *ptrListaFotogramasClaves* cada fotograma clave, y sea procesado para obtener su evaluación. Esta evaluación es enviada a través del mecanismo *signals-slot* a la clase *Principal*, la que enviará la evaluación de la calidad al Monitor a través de la clase *IPlugin*.

Las clases *hiloCapturarStreaming*, *hiloDeterminarFotogramasClaves* e *hiloDeterminarCalidadVideo*, representan hilos de procesamiento, diseñado de esta manera para minimizar el tiempo de respuesta y lograrse la evaluación en tiempo real. Estas clases hacen uso de la biblioteca *OpenCV* para la captura del flujo de video y el procesamiento de las imágenes.

Todo este proceso se ejecutará hasta que el monitor detenga la transmisión o la propia evaluación.

2.7 Modelo de implementación

El Modelo de Implementación es comprendido por un conjunto de componentes que constituyen la composición física de la implementación del sistema, describe cómo se organizan los componentes en dependencia de la estructura que posean en su entorno de implementación y las tecnologías seleccionadas para su desarrollo. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Además, describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (Hernández, 2013).

2.7.1 Diagrama de componentes

Muestra cómo el sistema está dividido en componentes, las dependencias entre ellos, las comunicaciones y localizaciones, las cuales son usadas para estructurar los componentes en los sistemas del software. Proveen una vista arquitectónica de alto nivel del sistema. Ayuda a los desarrolladores a visualizar el camino de la implementación. Permite tomar decisiones respecto a las tareas de implementación. Describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Representan todos los tipos de elementos de *software* que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente (Alba, 2011).

A continuación se muestra el diagrama de componentes desarrollado para representar los elementos de *software* presentes en la construcción de la aplicación:

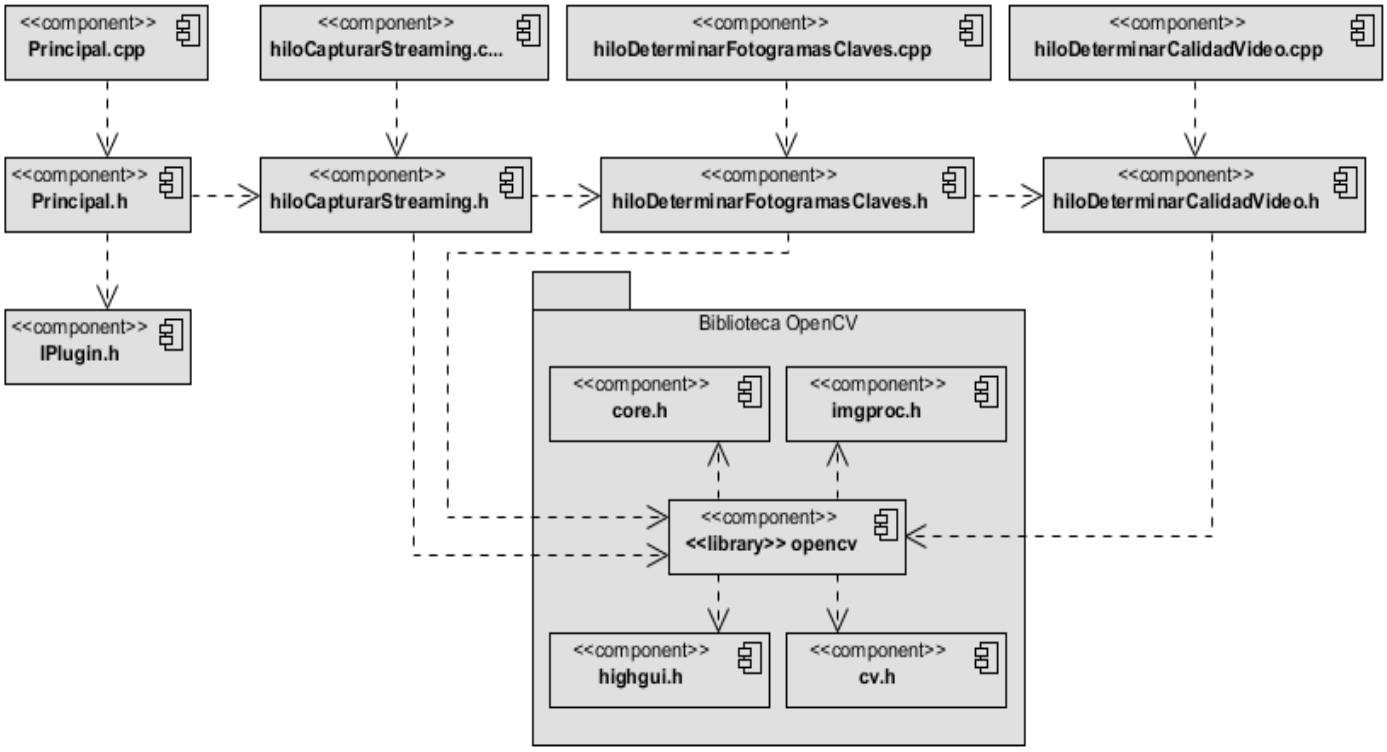

Fig 4: Diagrama de componentes.

2.7.2 Modelo de despliegue

Este modelo es utilizado como entrada fundamental en las actividades de diseño e implementación (Jacobson, y otros, 1999). Los Diagramas de Despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene algo de memoria y, a menudo, capacidad de procesamiento. Los nodos se utilizan para modelar la topología del hardware sobre el que se ejecuta el sistema. Representa típicamente un procesador o un dispositivo sobre el que se pueden desplegar los componentes (Alba, 2011).

Para representar la distribución de los elementos del componente desarrollado, a continuación se presenta el diagrama de despliegue:

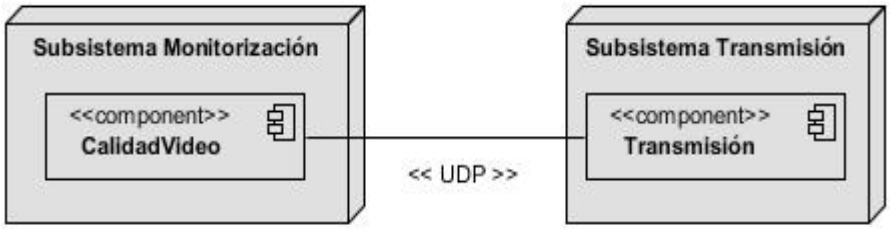

Fig 5: Diagrama de despliegue.

2.8 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. También se puede entender como un conjunto de reglas de notación y nomenclatura, específicas de cada lenguaje de programación, que se usan y se siguen durante la fase de implementación de una aplicación y reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores que no son detectados por lo compiladores, y reducen el tiempo y coste de las actividades de depuración y pruebas necesarias para la detección y corrección de los mismos (Remedio, y otros, 2007).

La UCI tiene definido un estándar de codificación para el lenguaje de programación C++, de esta manera el proyecto STCV utiliza dicho estándar para el desarrollo de sus productos. En la presente investigación se hace uso de este estándar, respetando así las políticas del proyecto, haciendo del resultado final, un componente que posea la estructura adecuada para facilitar la lectura, comprensión y mantenimiento del código en el sistema.

A continuación se muestran algunas de las características con las cuales cumple la implementación del componente:

Nombres de identificadores:

- Los atributos deben comenzar con letra minúsculas.
- Los parámetros deben comenzar con la letra p y con nombre lo más similar posible al atributo que se refiere.

Identificadores de variables:

- Para distinguir palabras dentro del nombre deberá emplearse una letra mayúscula.

- Para los atributos que representan listas, el nombre de éste comenzará con la palabra lista, seguida por un nombre sugerente.

Identificadores de punteros (apuntadores):

- Su nombre deberá comenzar con el prefijo ptr.

Identificadores de funciones:

- La primera letra deberá ser mayúscula.
- En caso de tener más de una palabra se distinguirán comenzando con letra mayúscula.

Generales:

- No manejar en los programas más de una instrucción por línea.
- Añadir comentarios descriptivos en cada declaración de variable, si es necesario.

2.9 Conclusiones parciales

Con la realización de este capítulo queda plasmada una propuesta inicial de cómo se realizará el desarrollo del componente de evaluación de la calidad de video digital durante las transmisiones del STCV. El modelo de dominio realizado permite una mejor comprensión del entorno donde radica el problema planteado al inicio de la investigación. La identificación de los requerimientos funcionales y no funcionales ofrece una visión clara y detallada de las condiciones y características que necesita el componente para el cumplimiento del objetivo de su desarrollo.

Se expuso el camino a seguir con el apoyo de la arquitectura elaborada, a partir de las condiciones del componente, el que posibilita una mayor comprensión de dicha arquitectura. Los patrones de diseño identificados guiarán todo el proceso de desarrollo de *software* y servirán como guía durante la implementación del componente.

La selección del estándar de codificación facilitará una mejor comprensión del código, gran utilidad para estudios y análisis posteriores del mismo.

Con la realización del modelo de implementación se describieron los componentes que intervienen en la construcción de la solución final y la distribución física que tendrán al funcionar como un todo dentro del componente de evaluación.

Capítulo 3: Validación y pruebas realizadas al componente

En el presente capítulo se pretende demostrar la validez de la presente investigación, con la ayuda de experimentos aplicados a cada una de las características seleccionadas para evaluar la calidad de un video digital. Luego de la aplicación de estos experimentos, se podrá validar el resultado de la investigación con la recopilación de los resultados alcanzados, que estará guiado además por medidas de eficiencia y efectividad que complementan las pruebas realizadas.

3.1 Descripción de la propuesta de solución

El componente de evaluación de calidad de video sobre las transmisiones del STCV, comienza el proceso cuando el monitor del subsistema Monitorización envía la url de la transmisión que va a ser evaluada, por medio de la clase *IPlugin*, que permite la comunicación entre el componente *CalidadVideo* y el sistema.

La función *CapturarStreaming* recibe la *url* y comienza a capturar el flujo de video, reescalando cada fotograma a 320x240 (largo x ancho) y almacenándolos con tipo de dato *Mat* en una lista (*ptrListaFotogramas*).

Simultáneamente a la función *CapturarStreaming*, se ejecuta la función *DeterminarFotogramasClaves*. Esta función, analiza cada uno de los fotogramas almacenados en la lista *ptrListaFotogramas*, seleccionando un fotograma por cada cambio de escena, almacenándolo en una lista (*ptrListaFotogramasClaves*).

Para determinar cuándo ocurre un cambio de escena, se realizan los siguientes pasos:

- Convertir formato de imagen de *Mat* a *IplImage*.
- Convertir la imagen *IplImage* de sistema de color *RGB* a escala de grises.
- Calcular el histograma de la imagen en escala de grises.

Si la variación del histograma de la imagen anterior en comparación con la actual es igual o superior al 10% atendiendo a la distribución de niveles de grises, se considera que hay un cambio de escena.

Simultáneamente a las 2 funciones anteriores, se ejecuta la función *ProcesarCaracteristicasVideo*, que accede al primer fotograma de la lista *ptrListaFotogramasClaves*, y ejecutará las funciones *DeterminarBrillo*, *DeterminarColor*, *DeterminarSaturacion* y *DeterminarContraste*, pasándole a las mismas los parámetros correspondientes para el procesamiento.

Esta función convierte el fotograma a evaluar, del tipo de dato *Mat* a *IplImage*, y luego lo convierte del sistema de color *RGB* a escala de grises. Además de ello, calcula el tamaño en píxeles del fotograma. Todo esto se hace con el objetivo de preparar los parámetros que se pasarán a las funciones que se ejecutarán posteriormente.

Se ejecuta la función *DeterminarBrillo*, que recibe por parámetro el fotograma en formato *IplImage* y sistema de color en escala de grises, además del tamaño de dicho fotograma. La función calcula el histograma para el fotograma recibido por parámetro, y analiza el porciento de píxeles de la imagen que sean oscuros y luminosos. Para ello, el histograma se crea con 16 celdas, resultando que los píxeles contenidos en las primeras 5 celdas, se consideran oscuros. Por otra parte, los píxeles contenidos en las últimas 6 celdas del histograma, se consideran luminosos. En caso que el porciento de uno de ellos sea igual o superior a 70, se considerará imagen con brillo oscuro o luminoso respectivamente. De no cumplirse alguno de esos casos, la imagen se considera con el brillo bien proporcionado. Insertando la evaluación correspondiente en una lista (*listaEvaluaciones*).

Se ejecuta la función *DeterminarColor*, que recibe por parámetro el fotograma en formato *IplImage* y sistema de color en escala de grises, además del tamaño de dicho fotograma. La función calcula el histograma para el fotograma recibido por parámetro, y analiza si se evidencia un gran predominio de un único color en la imagen. Para ello, se crea el histograma con 8 celdas, analizándose el porciento de píxeles que contiene cada celda con respecto a la cantidad de píxeles de la imagen. Si en algún caso, el porciento es igual o superior a 50, se considera que la imagen tiene gran predominio de un color. En otro caso, se considera la imagen con colores bien proporcionados. Finalmente se inserta la evaluación correspondiente en *listaEvaluaciones*.

Se ejecuta la función *DeterminarSaturacion*, que recibe por parámetro el fotograma en formato *Mat* y sistema de color *RGB*, además del tamaño de dicho fotograma. La función convierte el fotograma a sistema de color *HSV*, para analizar la 2da componente de este sistema de color, la Saturación. Luego, se accede a la información de cada píxel, analizándose si el mismo está saturado (un píxel se considera saturado si su valor de saturación excede el valor 146, 146 representa las 4/7 partes de 256). Posteriormente se analiza el porcentaje de píxeles saturados en la imagen, y en caso de ser igual o superior a 70, se considera la imagen altamente saturada, en caso de que dicho valor sea igual o inferior a 10, se considera la imagen poco saturada, y en otro caso, se considera la imagen con niveles de saturación bien proporcionados. Finalmente se inserta la evaluación correspondiente en *listaEvaluaciones*.

Se ejecuta la función *DeterminarContraste*, que recibe por parámetro el fotograma en formato *IplImage* y sistema de color en escala de grises. La función calcula el histograma para el fotograma recibido por parámetro, y analiza el porciento de niveles de grises que están presentes en la imagen (un nivel de gris se considera presente en la imagen cuando al menos 300 píxeles tienen ese nivel de gris). En caso que el porciento sea igual o superior a 40, se considerará imagen con contraste, en otro caso, se considerará imagen con poco contraste. Finalmente se inserta la evaluación correspondiente en *listaEvaluaciones*.

Una vez ejecutadas todas las funciones anteriormente descritas por cada uno de los fotogramas claves, se retornará la lista de evaluaciones (*listaEvaluaciones*) al monitor, haciendo uso de la clase *IPlugin*, posibilitando la evaluación de cada uno de los fotogramas claves en tiempo real.

3.2 Experimentos realizados al componente

Los experimentos se realizan sobre las características de la imagen que fueron procesadas, con el fin de analizar si la investigación ofrece los resultados esperados. Para esto se definen medidas de eficiencia que en la estadística, pretenden resumir la información de la muestra para poder tener así un mejor conocimiento de la población. En ella, los términos falso positivo (FP) y falso negativo (FN) son usados para describir posibles errores.

Cuando se realiza evaluación de la calidad sobre imágenes, se produce un FP cuando se emite una evaluación de la imagen, sin embargo esa evaluación no se cumple. Por otro lado, se produce un FN, cuando se concluye que una imagen no tiene una evaluación, sin embargo la imagen presenta dicha evaluación. Ambas clasificaciones permiten calcular la eficacia del componente (Boqué, 2008).

Análogamente, se define el término evaluación correcta (EC) para describir las decisiones correctas del proceso de evaluación. Una EC se produce cuando se evalúa correctamente la característica que se analiza de la imagen. Este término se utiliza en caso que el experimento se vaya a realizar con clasificadores no binarios. En caso de clasificadores binarios, se emplearán los términos de verdaderos positivos (VP) y verdaderos negativos (VN) en sustitución de EC. Se emplea EC como una adaptación de los términos usados para describir los resultados de un experimento binario, debido a que es necesario realizar los mismos con más de dos variables.

Para un mejor entendimiento de lo abordado, se presenta la siguiente tabla, definida sobre el contraste de la imagen:

Tabla 2: Términos usados para describir los resultados del experimento sobre el contraste de la imagen.

		Imagen con contraste (ICC)	Imagen con poco contraste (IPC)
Resultado obtenido	Positivo	ICC + Positivo = VP	IPC + Positivo = FP
	Negativo	ICC + Negativo = FN	IPC + Negativo = VN

La eficiencia máxima se consigue cuando el componente es capaz de procesar un conjunto amplio de imágenes sin producir ningún falso positivo y ningún falso negativo. Esto se traduce en que las evaluaciones de las imágenes se realizaron correctamente al 100%. En la literatura existen diversas técnicas para medir la eficiencia de estos algoritmos, aunque una de ellas destaca sobre las demás por su amplia utilización. Se trata de las medidas de *recall* y *precision* definidas como:

Para clasificaciones binarias:

$$Recall = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FN}}$$

$$Precision = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FP}}$$

Para clasificaciones no binarias:

$$Recall = \frac{\mathrm{EC}}{\mathrm{EC} + \mathrm{FN}}$$

$$Precision = \frac{\mathrm{EC}}{\mathrm{EC} + \mathrm{FP}}$$

Recall expresa la razón de clasificaciones de las imágenes correctamente detectadas entre todas las existentes, mientras que *precision* ofrece una medida de la fiabilidad de la detección, que significa cuántas de las imágenes detectadas son realmente correctas y no falsos positivos. Estas medidas normalmente se trabajan en el dominio [0,1], sin embargo se pueden utilizar en forma porcentual.

Para medir la calidad de una imagen se utilizan varias clasificaciones relacionadas con la percepción visual humana sobre las características evaluadas.

3.2.1 Experimentos sobre el brillo

Para la evaluación de la característica brillo de una imagen, se convierte la imagen a escala de grises, y se calcula su histograma en 16 celdas. Para establecer clasificaciones sobre el histograma, se analizaron

sus posiciones tomando las primeras 5 celdas para calcular el porciento de oscuridad presente en la imagen, y las últimas 6 para calcular el de luminosidad. Según los valores porcentuales obtenidos, se establece una evaluación.

Se analizó una muestra de 200 imágenes oscuras, 200 luminosas y 200 con brillo bien distribuido. Como se muestra en la tabla, los resultados más acertados en la detección de imágenes oscuras y luminosas estuvieron dados sobre los valores iguales o superiores a 70%, mientras que los resultados más acertados en la detección de imágenes con buena distribución de brillo estuvieron dados sobre los valores inferiores a 70%.

Tabla 3: Resultados obtenidos sobre una muestra de imágenes oscuras y luminosas.

	Valores porcentuales									
Resultados obtenidos	**≥0**	**≥10**	**≥20**	**≥30**	**≥40**	**≥50**	**≥60**	**≥70**	**≥80**	**≥90**
Oscuras	0	0	0	0	0	0	0	46	64	90
Luminosas	0	0	0	0	0	0	0	50	89	61

Al evaluar una imagen en cuanto al brillo, si no cumple con los porcientos anteriormente establecidos de oscuridad y luminosidad, se considera que la imagen posee una buena distribución de brillo.

3.2.2 Experimentos sobre el color

Para la evaluación de la característica color de una imagen, se convierte la imagen a escala de grises, y se calcula su histograma en 8 celdas. Posteriormente, se calcula el porciento de píxeles que contiene cada celda con respecto a la cantidad total de píxeles de la imagen.

Se analizó una muestra de 200 imágenes donde predomina algún color y 200 imágenes con colores bien distribuidos. Como se muestra en la tabla, los resultados más acertados en la detección de imágenes donde predomina algún color estuvieron dados sobre los valores iguales o superiores a 50%, mientras que, los resultados más acertados en la detección de imágenes con color bien distribuidos estuvieron dados sobre los valores inferiores a 50%.

Tabla 4: Resultados obtenidos sobre una muestra de imágenes con predominio de un color y bien distribuidas.

	Valores porcentuales

Resultados obtenidos	≥0	≥10	≥20	≥30	≥40	≥50	≥60	≥70	≥80	≥90
Predominio de un color	0	0	0	0	0	45	43	45	40	27
Color bien distribuido	0	25	107	62	6	0	0	0	0	0

3.2.3 Experimentos sobre la saturación

Para la evaluación de la característica saturación de una imagen, fue necesario convertir la imagen al sistema de color HSV, donde la componente S representa la saturación. Luego se analiza cada píxel de la imagen, identificando su nivel de saturación. Se considera que un píxel está saturado cuando su componente S excede el valor 146, donde 146 representa las 4/7 partes de 256, que es la cantidad de niveles de saturación que contiene una imagen HSV. Para establecer una evaluación de saturación, se calcula el porciento de píxeles saturados que contiene la imagen.

Se analizó una muestra de 200 imágenes con alta saturación, 200 con baja saturación y 200 con buena saturación. Como se muestra en la tabla, los resultados más acertados en la detección de imágenes con alta saturación estuvieron dados sobre los valores iguales o superiores a 70%, mientras que, los resultados más acertados en la detección de imágenes con baja saturación estuvieron dados sobre los valores iguales o inferiores a 10%.

Tabla 5: Resultados obtenidos sobre una muestra de imágenes con alta y baja saturación.

	Valores porcentuales									
Resultados obtenidos	**≥0**	**≥10**	**≥20**	**≥30**	**≥40**	**≥50**	**≥60**	**≥70**	**≥80**	**≥90**
Altamente saturadas	0	0	0	0	0	0	1	64	59	76
Baja saturación	200	0	0	0	0	0	0	0	0	0

Al evaluar una imagen en cuanto a la saturación, si no cumple con los porcientos anteriormente establecidos de alta y baja saturación, se considera que la imagen posee una buena distribución de la misma.

3.2.4 Experimentos sobre el contraste

Para la evaluación de la característica contraste de una imagen, se convierte la imagen a escala de grises, y se calcula su histograma en 256 celdas. Teniendo en cuenta que la imagen cuando se captura es

reescalada a 320x240 y que su cantidad total de píxeles es 76800, se concluye que en cada nivel de gris debe haber 300 píxeles para que la imagen resulte contrastada. Posteriormente, se calcula el porciento de niveles de gris que contienen al menos 300 píxeles.

Se analizó una muestra de 200 imágenes con contraste y 200 con poco contraste. Como se muestra en la tabla, los resultados más acertados en la detección de imágenes con contraste estuvieron dados sobre los valores iguales o superiores a 40%, mientras que, los resultados más acertados en la detección de imágenes con poco contraste estuvieron dados sobre los valores inferiores a 40%.

Tabla 6: Resultados obtenidos sobre una muestra de imágenes con contraste y poco contraste.

	Valores porcentuales									
Resultados obtenidos	**≥0**	**≥10**	**≥20**	**≥30**	**≥40**	**≥50**	**≥60**	**≥70**	**≥80**	**≥90**
Con contraste	0	0	0	0	57	80	33	22	6	2
Con poco contraste	14	46	60	80	0	0	0	0	0	0

3.3 Resultados obtenidos

Luego de haber identificado los umbrales para la evaluación de las características de una imagen, se procede a realizar experimentos sobre una muestra de 1000 imágenes extraídas de una transmisión del STCV, para evaluar según los umbrales definidos y la percepción visual humana. Para el experimento se contó con la participación de 5 profesionales del proyecto STCV para que emitieran una evaluación según su percepción visual, y esta evaluación se comparó con lo arrojado por los umbrales definidos, estos resultados se plasman en la siguiente tabla.

Las clasificaciones que se muestran en las tablas siguientes se otorgaron en dependencia de un umbral definido para cada indicador, obtenido con la realización de experimentos.

Para evaluar los resultados de este experimento, se utilizaron los conceptos relacionados en las tablas siguientes y las métricas de eficiencia definidas con anterioridad.

3.3.1 Validación del brillo

Tabla 7: Resultados de brillo obtenidos de una muestra de imágenes aleatorias.

	Valores porcentuales	**Medidas de eficiencia**

Clasificaciones	EC	FP	FN	*Recall*	*Precision*
Oscura	302	2	0	1	0.99
Luminosa	48	1	0	1	0.98
Buena	632	15	0	1	0.98

Como se puede apreciar, en los resultados arrojados para *recall* y *precision* se obtuvieron valores muy cercanos a 1, por lo que se concluye que los resultados alcanzados son satisfactorios.

3.3.2 Validación del color

Tabla 8: Resultados de color obtenidos de una muestra de imágenes aleatorias.

Resultados del reconocimiento				Medidas de eficiencia	
VP	VN	FP	FN	*Recall*	*Precision*
965	14	21	0	1	0.98

Como se puede apreciar, en los resultados arrojados para *recall* y *precision* se obtuvieron valores muy cercanos a 1, por lo que se concluye que los resultados alcanzados son satisfactorios.

3.3.3 Validación de la saturación

Tabla 9: Resultados de saturación obtenidos de una muestra de imágenes aleatorias.

	Valores porcentuales			Medidas de eficiencia	
Clasificaciones	**EC**	**FP**	**FN**	***Recall***	***Precision***
Alta saturación	43	1	0	1	0.98
Baja saturación	126	1	0	1	0.99
Buena saturación	816	13	0	1	0.98

Como se puede apreciar, en los resultados arrojados para *recall* y *precision* se obtuvieron valores muy cercanos a 1, por lo que se concluye que los resultados alcanzados son satisfactorios.

3.3.4 Validación del contraste

Tabla 10: Resultados de contraste obtenidos de una muestra de imágenes aleatorias.

Resultados del reconocimiento				**Medidas de eficiencia**	
VP	VN	FP	FN	*Recall*	*Precision*
947	23	30	0	1	0.97

Como se puede apreciar, en los resultados arrojados para *recall* y *precision* se obtuvieron valores muy cercanos a 1, por lo que se concluye que los resultados alcanzados son satisfactorios.

3.4 Prueba de software

Según (Pressman, 2010) realizar una prueba de software en la etapa de prueba, consiste en que el ingeniero de software crea una serie de casos de prueba que pretenden demoler el software construido. Las pruebas son la actividad en la cual se somete a un sistema o uno de sus componentes a una evaluación de los resultados que arroja en base a la ejecución de éste en condiciones especificadas. *"Una prueba puede demostrar la presencia de errores pero no la ausencia de ellos"* según E. W. Dijkstra. Las pruebas de software se ubican principalmente en la evaluación de la calidad del producto y representan un elemento crítico para la garantía del mismo. Las pruebas no se deben realizar sólo al concluir el ciclo de desarrollo del software, sino durante todo su ciclo de vida, pues aunque realizar pruebas a un sistema informático no significa que el proceso de desarrollo esté asegurado, realizarlo formalmente y en tiempo es un buen inicio para aumentar su calidad final (Pressman, 2010).

3.4.1 Pruebas Unitarias

La prueba de unidad se centra en el componente. Usando la descripción del diseño detallado como guía, se prueban los caminos de control importantes con el fin de descubrir errores dentro del componente. La prueba de unidad hace uso intensivo de las técnicas de prueba de caja blanca (Jiménez, y otros, 2012). Primero se prueban los bloques desarrollados más pequeños del programa, antes de probar el software en su totalidad y así facilita la tarea de eliminar errores (Myers, 2004). El procedimiento para el diseño de casos de prueba para una prueba de unidad es el siguiente: analizar la lógica del componente usando uno de los métodos de caja blanca y después completar los casos de prueba aplicando métodos de caja negra a la especificación del componente (Myers, 2004).

Pruebas de caja blanca

La prueba del camino básico es una técnica de prueba de caja blanca. Este método permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del problema (Pressman, 2010).

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica del componente. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico del componente y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Pressman, 2010). Un camino independiente es cualquier camino del componente que introduce un nuevo conjunto de sentencias de proceso o una nueva condición. A continuación se listan los pasos a seguir para la realización de las pruebas de caja blanca:

1. Generar el grafo de flujo de procesamiento.

2. Calcular la complejidad ciclomática V (G), de un grafo de flujo G.

    V (G) = Número de Aristas (NA) - Número de Nodos (NN) + 2.

3. Determinar los caminos independientes o básicos.

4. Generar un caso de prueba para cada camino de ejecución.

Diseño de prueba de caja blanca

Las pruebas de caja blanca permiten comprobar que se cumplan todos los caminos lógicos del componente mediante la aplicación de la técnica del camino básico. En un grafo puede haber un número pequeño de caminos, pero también puede existir un número infinito de ellos. Para solucionar este problema existe un método de prueba llamado prueba estructurada, que consiste en probar conjuntos de caminos que formen una base para todos los posibles caminos en el grafo. Esta prueba utiliza dos variantes: método simplificado y método general (Pressman, 2010).

Para la obtención de los caminos básicos de las funciones del componente se utilizará el método general. En este método se elige el primer camino como uno que tenga sentido funcional y represente la operación normal del componente. A partir de ese camino inicial, se van obteniendo otros hasta completar los

caminos necesarios. A continuación se exponen las funciones del componente seleccionadas para estas pruebas:

Función *exec()* de la clase *hiloDeterminarFotogramasClaves* (Ver Anexo 1)

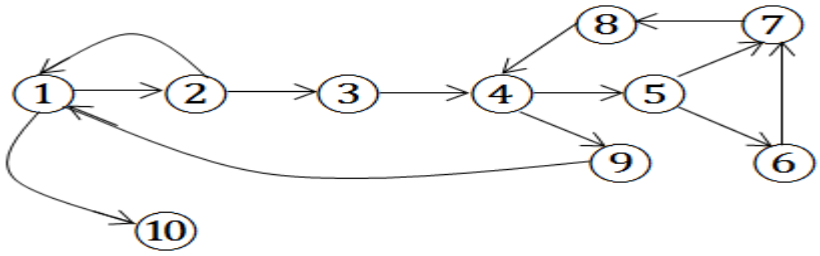

Fig 6: Grafo de la función *exec* de la clase *hiloDeterminarFotogramasClaves*.

Tabla 11: Caso de prueba para los caminos básicos de la función *exec*.

Camino básico	Función del código	Resultado
1-2-3-4-5-6-7-8-4-9-1-10	Se ejecuta el proceso concurrente. La lista de fotogramas no está vacía. Se bloquea la función *InsertarFotograma*. Se analizan todos los fotogramas de la lista, no ocurre un cambio de escena. Se limpia la lista de fotogramas. Se desbloquea la función *InsertarFotograma*. No se ejecuta el proceso concurrente. Termina la función.	Satisfactorio
1-2-3-4-5-7-8-4-9-1-10	Se ejecuta el proceso concurrente. La lista de fotogramas no está vacía. Se bloquea la función *InsertarFotograma*. Se analizan todos los fotogramas de la lista, si ocurre un cambio de escena se actualiza la lista de fotogramas claves. Se limpia la lista de fotogramas. Se desbloquea la función *InsertarFotograma*. No se ejecuta el proceso concurrente. Termina la función.	Satisfactorio
1-2-3-4-9-1-10	Se ejecuta el proceso concurrente. La lista de fotogramas no está vacía. Se bloquea la función *InsertarFotograma*. Se limpia la lista de fotogramas. Se desbloquea la función *InsertarFotograma*. No se ejecuta el proceso concurrente. Termina la función.	Satisfactorio
1-10	No se ejecuta el proceso concurrente. Termina la función.	Satisfactorio

1-2-1-10	Se ejecuta el proceso concurrente. La lista de fotogramas está vacía. No se ejecuta el proceso concurrente. Termina la función.	Satisfactorio

Función *DeterminarBrillo()* de la clase *hiloDeterminarCalidadVideo* (Ver Anexo 2)

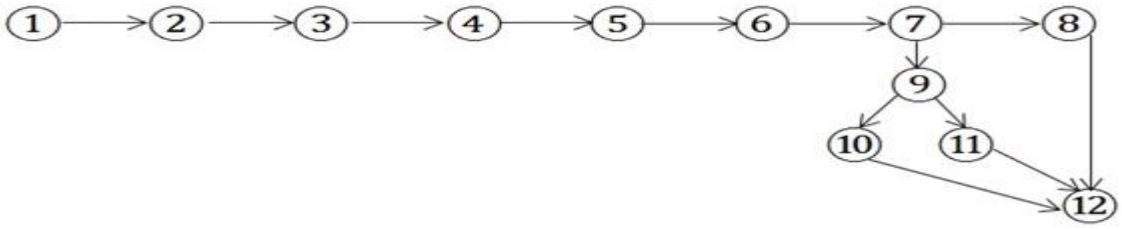

Fig 7: Grafo de la función *DeterminarBrillo* de la clase *hiloDeterminarCalidadVideo*.

Tabla 12: Caso de prueba para los caminos básicos de la función *DeterminarBrillo*.

Camino básico	Función del código	Resultado
1-2-3-2-4-5-6-5-7-8-12	Se declaran y asignan valores a las variables. Se obtiene la cantidad de píxeles oscuros y luminosos y se calculan los porcientos que éstos representan en la imagen. Se evalúa la imagen como oscura. Termina la función.	Satisfactorio
1-2-3-2-4-5-6-5-7-9-10-12	Se declaran y asignan valores a las variables. Se obtiene la cantidad de píxeles oscuros y luminosos y se calculan los porcientos que éstos representan en la imagen. Se evalúa la imagen como luminosa. Termina la función.	Satisfactorio
1-2-3-2-4-5-6-5-7-9-11-12	Se declaran y asignan valores a las variables. Se obtiene la cantidad de píxeles oscuros y luminosos y se calculan los porcientos que éstos representan en la imagen. Se evalúa la imagen con buena distribución de brillo. Termina la función.	Satisfactorio
1-2-3-2-4-5-6-7-8-12	Se declaran y asignan valores a las variables. Se obtiene el total de píxeles oscuros, no se obtiene el	Insatisfactorio, debido a que este camino en el código nunca

	total de píxeles luminosos.	se ejecuta. Siempre se obtiene el total de píxeles luminosos.
1-2-3-4-5-6-5-7-8-12	Se declaran y asignan valores a las variables. No se obtiene el total de píxeles oscuros.	Insatisfactorio, debido a que este camino en el código nunca se ejecuta. Siempre se obtiene el total de píxeles oscuros.
1-2-3-2-4-5-6-5-7-8-12	Se declaran y asignan valores a las variables. Se obtiene la cantidad de píxeles oscuros y luminosos y se calculan los porcientos que éstos representan en la imagen. Se evalúa la imagen como oscura. Termina la función.	Satisfactorio

Función *DeterminarColor()* de la clase *hiloDeterminarCalidadVideo* (Ver Anexo 3)

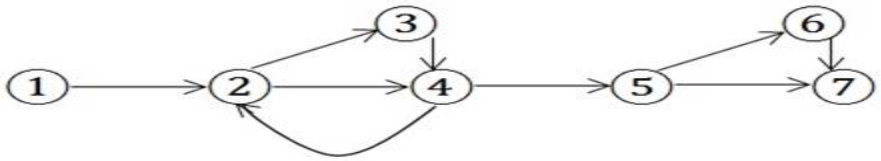

Fig 8: Grafo de la función *DeterminarColor* de la clase *hiloDeterminarCalidadVideo*.

Tabla 13: Caso de prueba para los caminos básicos de la función *DeterminarColor*.

Camino básico	Función del código	Resultado
1-2-3-4-5-7	Se declaran y asignan valores a las variables. Se obtiene la cantidad de píxeles del mismo color y se calculan los porcientos que éstos representan en la imagen. Se evalúa la imagen con predominio de un sólo color. Termina la función.	Satisfactorio
1-2-4-2-4-5-6-7	Se declaran y asignan valores a las variables. Se obtiene la cantidad de píxeles del mismo color y se calculan los porcientos que éstos representan en la imagen. Se evalúa la imagen con buena distribución. Termina la función.	Satisfactorio
1-2-4-5-7	Se declaran y asignan valores a las variables. Se obtiene la	Insatisfactorio, debido a

	cantidad de píxeles del mismo color y se calculan los porcientos que éstos representan en la imagen. No se obtiene una evaluación del color de la imagen. Termina la función.	que este camino en el código nunca se ejecuta. Siempre se obtiene una evaluación del color.
1-2-4-2-3-4-5-7	Se declaran y asignan valores a las variables. Se obtiene la cantidad de píxeles del mismo color y se calculan los porcientos que éstos representan en la imagen. Se evalúa la imagen con predominio de un sólo color. Termina la función.	Satisfactorio

Función *DeterminarSaturacion()* de la clase *hiloDeterminarCalidadVideo* (Ver Anexo 4)

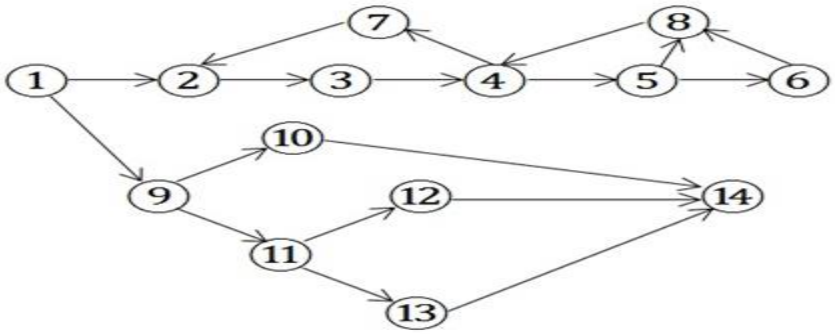

Fig 9: Grafo de la función *DeterminarSaturacion* de la clase *hiloDeterminarCalidadVideo*.

Tabla 14: Caso de prueba para los caminos básicos de la función *DeterminarSaturacion*.

Camino básico	Función del código	Resultado
1-2-3-4-5-6-8-4-7-2-9-10-14	Se declaran y asignan valores a las variables. Se obtiene el valor de saturación de cada píxel de la imagen, se analiza si está saturado y se incrementa la cantidad de píxeles saturados. Se analiza el porciento de píxeles saturados que contiene la imagen. Se evalúa la imagen de alta saturación. Termina la función.	Satisfactorio
1-2-3-4-5-6-8-4-7-2-9-11-12-14	Se declaran y asignan valores a las variables. Se obtiene el valor de saturación de cada píxel de la imagen, se analiza si está saturado y se incrementa la cantidad de píxeles	Insatisfactorio, debido a que este camino básico nunca se ejecutará, ya

	saturados. Se analiza el porciento de píxeles saturados que contiene la imagen. Se evalúa la imagen de baja saturación. Termina la función.	que el único píxel de la imagen que se analiza está saturado, por tanto la imagen se evaluará de alta saturación.
1-2-3-4-5-6-8-4-7-2-9-11-13-14	Se declaran y asignan valores a las variables. Se obtiene el valor de saturación de cada píxel de la imagen, se analiza si está saturado y se incrementa la cantidad de píxeles saturados. Se analiza el porciento de píxeles saturados que contiene la imagen. Se evalúa la imagen de buena saturación. Termina la función.	Insatisfactorio, debido a que este camino básico nunca se ejecutará, ya que el único píxel de la imagen que se analiza está saturado, por tanto la imagen se evaluará de alta saturación.
1-2-9-10-14	Se declaran y asignan valores a las variables. No se analiza el nivel de saturación de los píxel de la imagen.	Insatisfactorio, debido a que no se analiza ningún píxel de la imagen.
1-2-3-4-7-2-9-11-12-14	Se declaran y asignan valores a las variables. No se analiza el nivel de saturación de los píxel de la imagen.	Insatisfactorio, debido a que no se analiza ningún píxel de la imagen.
1-2-3-4-5-8-4-7-2-9-11-12-14	Se declaran y asignan valores a las variables. Se obtiene el valor de saturación de cada píxel de la imagen, se analiza si está saturado y no se incrementa la cantidad de píxeles saturados. Se analiza el porciento de píxeles saturados que contiene la imagen. Se evalúa la imagen de baja saturación. Termina la función.	Satisfactorio

Función *DeterminarContraste()* de la clase *hiloDeterminarCalidadVideo* (Ver Anexo 5)

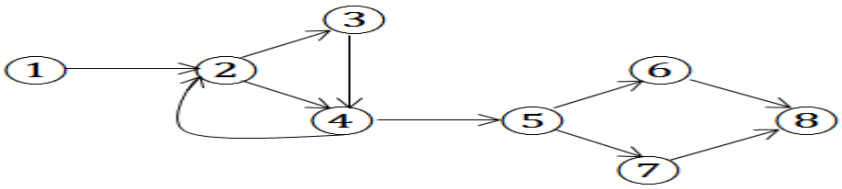

Fig 10: Grafo de la función *DeterminarContraste* de la clase *hiloDeterminarCalidadVideo*.

Tabla 15: Caso de prueba para los caminos básicos de la función *DeterminarContraste*.

Camino básico	**Función del código**	**Resultado**
1-2-3-4-2-4-5-6-8	Se declaran y asignan valores a las variables. Se obtiene la cantidad de píxeles que contiene cada nivel de gris en la imagen, se analiza y se incrementa la cantidad de niveles de grises presentes en la imagen. Se analiza el porciento de niveles de grises que contiene la imagen. Se evalúa la imagen con contraste. Termina la función.	Satisfactorio
1-2-3-4-2-4-5-7-8	Se declaran y asignan valores a las variables. Se obtiene la cantidad de píxeles que contiene cada nivel de gris en la imagen, se analiza y se incrementa la cantidad de niveles de grises presentes en la imagen. Se analiza el porciento de niveles de grises que contiene la imagen. Se evalúa la imagen de poco contraste. Termina la función.	Satisfactorio
1-2-4-5-6-8	Se declaran y asignan valores a las variables. Se obtiene y analiza la cantidad de píxeles que contiene solamente el primer nivel de gris en la imagen.	Insatisfactorio, debido a que en el código siempre se obtienen y analizan todos los niveles de grises de la imagen.
1-2-3-4-5-6-8	Se declaran y asignan valores a las variables. Se obtiene y analiza la cantidad de píxeles que contiene solamente el primer nivel de gris en la imagen.	Insatisfactorio, debido a que en el código siempre se obtienen y analizan todos los niveles de grises de la

		imagen.

3.4.2 Pruebas Integración

Son ejecutadas para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados. La necesidad de realizar las pruebas de integración viene dada por el hecho de que los módulos que forman un programa suelen fallar cuando trabajan de forma conjunta, aunque previamente se haya demostrado que funcionan correctamente de manera individual (Vence, 2009). Los errores más comunes guardan relación con la comunicación a través de las interfaces, acumulación notable de errores de cálculo, acceso incoherente a estructuras de datos globales, tiempos de respuestas, entre otros. Estas pruebas son posteriores a las pruebas unitarias.

Las principales dificultades que surgen en la pruebas de integración es localizar los errores que se descubren durante el proceso. Existen interacciones complejas entre los componentes del sistema, y cuando se descubre una salida anómala, es difícil encontrar la fuente del error. Para hacer más fácil la localización de errores, se utiliza un enfoque incremental para la integración y pruebas del sistema. De forma inicial, se debe integrar una configuración mínima del sistema y probar dicho sistema. Luego se agregan componentes a esta configuración mínima y se prueba después de cada incremento.

Para la realización de esta prueba, se utiliza la estrategia ascendente que especifica que los componentes de los niveles bajos se integran y prueban antes que se desarrollen los componentes de los niveles altos. Este enfoque no requiere que el diseño arquitectónico del sistema esté completo, por lo que se puede comenzar en una etapa inicial del proceso de desarrollo. Para la integración del componente de evaluación de la calidad de video con el subsistema Monitorización, se comienza incluyendo la interfaz *IPlugin* definida por el subsistema, para establecer la comunicación con los demás *plugins* que participen en la monitorización de las transmisiones. En la clase *Principal* se establece una herencia de la clase *IPlugin* y se re-implementa el método *Comunicacion*, declarado en la clase *IPlugin*. A través de esta interfaz el subsistema va a iniciar y detener los *plugins* con los que interactúa.

Luego de haber incluido la clase *IPlugin* como parte de la implementación del componente desarrollado, se procede a compilar todo el código para generar un archivo con extensión .so, que contiene encapsuladas las funciones del componente, copiando este archivo en el directorio que va a contener los *plugins* de evaluación de calidad de video de la monitorización. Finalmente se inicia el subsistema Monitorización y se procede a efectuar la evaluación de la calidad, verificando que el componente

funcione correctamente luego de la integración, emitiendo la evaluación de la calidad del video que se transmite.

En la fase de prueba se realizaron dos iteraciones al componente, en la primera iteración el componente funcionaba correctamente, pero el subsistema no estaba equipado para recibir las notificaciones del componente. El componente integrado no mostraba errores de implementación pero no fue posible visualizar el resultado. En la segunda iteración, el subsistema preparó su entorno para recibir y mostrar la notificación de la evaluación de la transmisión por parte del componente. Esta segunda fase concluyó de forma satisfactoria sin encontrar problemas de funcionamiento entre las partes integradas.

3.4.3 Pruebas Funcionales

Pruebas de caja negra

En las pruebas de software, conociendo una función específica para la que fue diseñado el producto, se pueden diseñar pruebas que demuestren que dicha función está bien realizada. Dichas pruebas son llevadas a cabo sobre las funciones del componente, actuando sobre ella como una caja negra, proporcionando unas entradas (el flujo de video) y estudiando las salidas para ver si son o no las esperadas. Estas pruebas examinan algunos aspectos del modelo del sistema, sin tener mucho en cuenta la estructura lógica interna del componente, y se centran en los requerimientos funcionales del software.

Caso de Prueba Evaluar calidad de video

Descripción general: El caso de uso (CU) inicia luego que el monitor ejecuta en el subsistema Monitorización la opción Evaluar calidad de video. Seguidamente el sistema captura el flujo de video y detecta los fotogramas claves para que sean procesados. Luego, se procede a evaluar el resultado del procesamiento. El CU termina cuando el sistema ha evaluado los resultados del procesamiento y emite una evaluación al monitor mediante el subsistema Monitorización.

Condiciones de ejecución: El subsistema Monitorización tiene una transmisión visualizándose.

SC Evaluar calidad de video

Tabla 16: Caso de Prueba Evaluar calidad de video.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1	El monitor del subsistema Monitorización comienza la	Muestra la evaluación en	STCV/Subsiste

Evaluar calidad de video.	evaluación de la calidad sobre una transmisión activa de un canal. Se captura la transmisión y se procesan los fotogramas claves que conforman el video que se transmite. **Observación:** Sólo se activará el botón Evaluar calidad en caso que exista una transmisión activa en un canal. Este botón no se podrá ejecutar nuevamente mientras no sea anteriormente detenida la evaluación.	tiempo real de la calidad atendiendo a las características: brillo, contraste, color y saturación de los fotogramas claves procesados.	ma Monitorización/ opción Evaluar Calidad.
EC 1.2 Evaluar calidad de video. Detener evaluación.	El monitor del subsistema Monitorización detiene la evaluación de la calidad sobre una transmisión activa de un canal. **Observación:** Sólo se activará el botón Detener evaluación en caso que se esté evaluando una transmisión en un canal. Este botón no se podrá ejecutar nuevamente mientras no sea anteriormente iniciada la evaluación.	Detiene la operación de evaluación de la calidad.	STCV/Subsiste ma Monitorización/ opción Detener Evaluación.

3.4.4 Pruebas de Rendimiento

Las pruebas de rendimiento son realizadas para comprobar qué tan rápido un sistema efectúa una tarea bajo ciertas circunstancias pre-planificadas de trabajo. Estas pruebas también son utilizadas para validar y verificar diferentes aspectos de la calidad de software.

Diseño de prueba de rendimiento

El componente de evaluación de calidad de video fue ejecutado en varias estaciones de trabajo, con el objetivo de demostrar su funcionamiento en tiempo real basándose en la rapidez de la respuesta de la evaluación emitida. El proceso se ejecutó en dos estaciones de trabajo del laboratorio 307 del docente 4, y una laptop personal, todas con prestaciones diferentes, lo que permitió identificar los requerimientos mínimos con los que funciona correctamente el componente desarrollado. A continuación se muestran los resultados obtenidos:

Tabla 17: Características de las estaciones de trabajo.

Estaciones	Prestaciones	Fotogramas procesados por segundo	Tiempo de respuesta (mseg[4])	Carga del CPU[5]	Carga de la RAM[6]
Puesto 1	Dual Core (2 CPUs 32 bits) ~ 2.2 GHz 2 GB RAM	34	29	90%	8.3%
Puesto 4	I3 (4 CPUs 64 bits) ~ 3.3 GHz 4 GB RAM	56	18	70%	4.2%
Laptop	i7 (8 CPUs 64 bits) ~ 3.5 GHz 8 GB RAM	60	16	60%	2.1%

El tiempo de respuesta es considerado como la cantidad de milisegundos que transcurren desde que se captura un fotograma hasta que se muestra su evaluación.

Con la realización de estas pruebas se demostró cómo el componente es capaz de funcionar con características diferentes de hardware, arrojando resultados satisfactorios en las tres estaciones que se utilizaron. Las pruebas evidenciaron la rapidez del procesamiento y con ello el envío de la evaluación en tiempo real, así como el buen tratamiento de los recursos de *hardware* de las computadoras, ya que se utiliza gran porciento del procesador y no se sobre carga la RAM, como consecuencia de la depuración realizada al código del componente, lo que resultaba de gran necesidad debido a que el procesamiento de imágenes es una tarea que consume muchos recursos.

3.5 Conclusiones parciales

[4] Milisegundos.

[5] Unidad Central de Procesamiento, del inglés *Central Processing Unit*.

[6] Memoria de Acceso Aleatorio, del inglés *Random Access Memory*.

Con la realización de los experimentos se pudieron validar los umbrales propuestos en el flujo de procesamiento para evaluar la calidad de video de una transmisión. Los resultados obtenidos mediante la aplicación de medidas de eficiencia demostraron la efectividad de la investigación. La realización de pruebas de software permitió validar la solución propuesta. Las mismas arrojaron como resultado que la aplicación cumple de manera satisfactoria, con las funcionalidades identificadas durante el proceso de desarrollo del software llevado a cabo como parte de la presente investigación. Las pruebas de caja blanca generaron casos de prueba que garantizaron que se ejecutaran por lo menos una vez todos los caminos independientes de las funciones que construye el componente desarrollado. Las pruebas de integración mostraron la convivencia del componente en el subsistema Monitorización, ofreciendo los resultados esperados al funcionar en conjunto. Las pruebas de rendimiento reflejaron la rapidez con la que funciona el componente bajo diferentes condiciones.

Conclusiones generales

Con la realización del presente trabajo de diploma se dio cumplimiento al objetivo general planteado al inicio de la investigación. El desarrollo de las tareas propuestas para la solución del problema científico permitió definir los aspectos teóricos y prácticos que construyeron la solución final. De este modo se puede concluir que:

- La utilización de RUP como metodología de desarrollo generó artefactos que representan una garantía para la continuidad del trabajo en el proyecto STCV.
- El lenguaje utilizado, las herramientas y biblioteca incluida ofrecieron el soporte necesario para lograr un producto con los requerimientos deseados, acordes con las exigencias planteadas por el cliente.
- El componente resultante fue estructurado siguiendo una arquitectura centrada en flujo de datos, con el apoyo de los estilos “Tuberías y Filtros” y “Llamada y Retorno”.
- Se definió un flujo de procesamiento para la evaluación de la calidad de video digital en transmisiones en tiempo real, el cual garantiza la monitorización de la calidad de la transmisión, disminuyendo el esfuerzo humano.
- Se identificaron umbrales sobre características de una imagen que fueron validados con la realización de experimentos sobre un conjunto amplio de imágenes. Los mismos demostraron la efectividad de la investigación, con la cual se sentaron las bases para trabajos futuros que mejoren estos resultados.
- La solución implementada agiliza y automatiza el chequeo de la calidad en el subsistema Monitorización.
- Se logró obtener un producto final fiable y de calidad avalado por las pruebas de software realizadas al componente.

De esta manera se puede concluir que se logró definir un flujo de procesamiento para la evaluación de la calidad de video, implementado como un componente que se integró al subsistema Monitorización luego de ser validado y probado para garantizar la efectividad del mismo.

Recomendaciones

Al haber cumplido los objetivos del presente trabajo y en correspondencia con los resultados obtenidos, se plantean las siguientes recomendaciones para trabajos futuros:

- Debido a la evolución que han tenido los métodos de evaluación objetiva de la calidad en los últimos años, se recomienda mantener un seguimiento de las posibles actualizaciones en los métodos SR, con el fin de poder adquirir en un futuro una solución patentada que permita una evaluación de la calidad sobre cualquier entorno.
- Por la complejidad lógica y práctica que posee la implementación de las distorsiones en videos en transmisión, se recomienda dedicar una investigación solamente a desarrollar la detección de una de las distorsiones, utilizando la técnica SR.

Bibliografía

Agustí, Manuel, Benlloch, Jose V. y Atienza, Vicente. 2010. *Adquisición y representación de medios digitales.* 2010.

Alba, Eduardo Rivera. 2011. *Arquitectura de Software II. Diagramas de Componentes y Despliegue.* Madrid. España : s.n., 2011.

Alfaro, Félix Murillo. 1999. Herramientas CASE. Instituto Nacional de Estadísticas e Informática. [En línea] Noviembre de 1999. http://www.inei.gob.pe/. 875-99-OI-OTDETI-INEI.

Barchiesi, Juan Vignolo. 2008. *Introducción al procesamiento digital de señales.* Chile : Ediciones Universitarias de Valparaíso, 2008. 978-956-17-0426-8.

Bass, L., Clements, P. y Kazman, R. 2003. *Software Architecture in Practice. 2nd Edition.* s.l. : Addison Wesley, 2003.

Bettstetter, Christian, Vögel, Hans-Jörg y Eberspächer, Jörg. 1999. *GSM PHASE 2, GENERAL PACKET RADIO SERVICE GPRS: ARCHITECTURE, PROTOCOLS,AND AIR INTERFACE.* Technische Universität München (TUM) : IEEE Communications Surveys, 1999.

Bezryadin, Sergey, Bourov, Pavel y Ilinih, Dmitry. 2013. *Brightness Calculation in Digital Image Processing.* San Francisco, CA, USA; UniquelC's, Saratov, Russia : KWE Int.Inc., 2013.

Boqué, Ricard. 2008. *EL LÍMITE DE DETECCIÓN DE UN MÉTODO ANALÍTICO.* s.l. : Tarragona : Universitat Rovira i Virgili, 2008.

Bustio, José Andrés Hernández. 2008. *Desarrollo del Canal Informativo del Ministerio del Poder Popular para la Energía y Petróleo de Venezuela: Subsistema de Trasmisión.* La Habana : s.n., 2008.

Casar, José R. 2005. *Tecnologías y servicios para la sociedad de la información.* Madrid, España : Universidad Politécnica de Madrid, 2005.

Castillo, Pedro Ángel Cuenca. 1999. *Codificación y transmisión robusta de señales de video MPEG-2 de caudal variables sobre redes de transmisión asíncrona ATM.* Castilla-La Mancha. España : Universidad de Castilla, 1999. 84-8427-013-0.

Chacón, Julio César Rueda. 2006. *Aplicación de la Metodología RUP para el desarrollo rápido de Aplicaciones Basado en el Estándar J2EE.* Guatemala : s.n., 2006.

Gamma, Erich, y otros. 2003. *Design Patterns.* 2003.

Giraldo, Gloria Lucia. 2009. *Actores y sus roles. Modelo del dominio.* Medellín. Colombia : s.n., 2009.

González, Sergio, Cheang León, Guillermo y Kashiwamoto Yabuta, Eiso Jorge. 2004. *Introducción a los sistemas distribuidos.* Ciudad de México : Universidad La Salle, 2004.

Hernández, Leovi Gilda. 2013. MODELO DE IMPLEMENTACIÓN. [En línea] 1 de junio de 2013. [Citado el: 21 de marzo de 2015.] http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html.

Hormigos, Carlos Esteban Baz. 2009. *MEDIDAS DE CALIDAD SUBJETIVA EN SECUENCIAS DE VIDEO.* Madrid : UNIVERSIDAD CARLOS III DE MADRID, 2009.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James . 1999. *El proceso unificado de desarrollo de software.* Madrid : s.n., 1999. 84-7829-036-2.

Jiménez, Omar y Rincón, Sebastián. 2012. *Pruebas de caja blanca y negra.* Colombia : Universidad Pedagógica y Tecnológica de Colombia, 2012.

Joskowicz, José y Sotelo, Rafael. 2012. Modelo de Estimación de Calidad de Video: Video Quality Experts Groups. 2012.

Katrib, Miguel. 1997. *Programación orientada a objetos en C++.* 1997.

Kazman, Rick. 2001. *Software Architecture. En Handbook of Software Engineering and Knowledge Engineering.* S-K Chang : World Scientific Publishing, 2001.

Larman, Craig. 2003. *UML y Patrones. Segunda Edición.* 2003.

León, Rolando Alfredo Hernández y González Coello, Sayda. 2011. *El proceso de investigación científica. .* La Habana : Editorial Universitaria del Ministerio de Educación Superior, 2011. 978-959-16-1307-3.

Lu Shi, I.K., Michael R. Lyu. 2011. Video Summarization Using Greedy Method in a Constraint Satisfaction Framework. [En línea] 2011. [Citado el: 12 de enero de 2015.] http://www.cse.cuhk.edu.hk/~lyu/paper_pdf/dms03.pdf..

Medina, Eduardo. 2014. Qt 5.2 supera el millón de descargas. [En línea] 17 de abril de 2014. [Citado el: 26 de noviembre de 2014.] http://www.muylinux.com/2014/04/17/qt-5-2-supera-el-millon-de-descargas.

Montagu, Arturo F. 2005. *CULTURA DIGITAL, COMUNICACIÓN Y SOCIEDAD.* Argentina : Paidós Ibérica, 2005. 9789501227208.

Montilva, Jonás A., Arapés, Nelson y Colmenares, Juan Andrés. 2003. *Desarrollo de software basado en componentes.* Mérida, Venezuela : s.n., 2003.

Myers, Glenford J. 2004. *The art of sotware Testing.* 2004.

OpenCV. 2003. The Open Computer Vision Library. . [En línea] 2003. http:www.intel.com/software/products/opensource/libraries/cvfl.htm.

Orallo, Enrique Hernández. 2003. El Lenguaje Unificado de Modelado (UML). . 2003.

Pinson, Margaret, y otros. 2002. *Video Quality Measurement PC .User's Manual.* 2002.

Pressman, Roger S. 2010. *Software Engineering. A Practitioner´s Approach. Seventh Edition.* New York. EE.UU : McGraw-Hill Companies, Inc., 2010. 978-0-07-337597-7.

Ramiro, Daniel Isaac Khan. 2011. *Interfaz gráfica multiplataforma para la simulación de ecuaciones físicas.* Madrid : Universidad Rey Juan Carlos, 2011.

Rao, K. R. y Hwang, J. J. 1996. *Techniques and Standards for Image, V'ideo and Audio Coding.* s.l. : Prentice Hall, 1996.

Remedio, Yaquelin y Román, Adisneisy C. 2007. *Propuesta de una guía para estandarizar la codificación en la Universidad.* La Habana : s.n., 2007.

Reynoso, Carlos y Kiccillof, Nicolás. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Versión 1.0.* Buenos Aires. Argentina : Universidad de Buenos Aires, 2004.

Shaw, Mary y Garlan, David. 1996. *SOFTWARE ARCHITECTURE: Perspectives on an Emerging Discipline.* s.l. : Prentice Hall - Upper Saddle River, N.J., 1996. 0-13-182957-2.

Sierra, María. 2006. *Trabajando con Visual Paradigm for UML. .* Cantabria : s.n., 2006.

Sommerville, Ian. 2000. *Software engineering.* s.l. : Addison-Wesley Pub Co, 2000.

Tovar, Lucia Barrera. 2010. *DESARROLLO E IMPLEMENTACIÓN DE ALGORITMOS PARA EL SISTEMA DE PERCEPCIÓN Y LOCALIZACIÓN DE LOS ROBOTS BOGOBOTS.* Atizapán de Zaragoza, Edo. México : INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY, 2010.

Vatolin, Dr. Dmitriy., y otros. 2010. MSU Video Quality Measurement Tool. *MSU Graphics & Media Lab (Video Group).* [En línea] 3 de 10 de 2010. [Citado el: 3 de Diciembre de 2014.] http://mmspg.epfl.ch/vqmt.

Vence, Jose María Pérez. 2009. *Plan de pruebas de integracion.* Madrid : s.n., 2009.

VQEG. 2008. Final Report of VQEG's Multimedia Phase I Validation Test. [En línea] 19 de Septiembre de 2008.

—. 2008. Objective perceptual multimedia video quality measurement . *Recommendation ITU-T J.247.* [En línea] Agosto de 2008.

—. 2008. Perceptual visual quality measurement techniques for multimedia services over digital cable television networks in the presence of a reduced bandwidth reference. *Recommendation ITU-T J.246, .* [En línea] Agosto de 2008.

—. 2009. *Validation of Reduced-Reference and No-Reference Objective Models for Standard Definition Television.* 2009.

—. 2011. Video Quality Experts Group. [En línea] Junio de 2011. [Citado el: 20 de Octubre de 2014.] http://www.its.bldrdoc.gov/vqeg/.

Wang, Lu Bovik. 2001. *Foveated Wavelet Image Quality Index.* 2001.

Woods, E Richard y González, Rafael C. 2002. *Digital Image Processing.* New Jersey : Prentice Hall, 2002.

Zielinski, Siegfried. 1999. *Audiovisions. Cinema and television as entr'actes in history.* Amsterdam : Amsterdam University Press, 1999.

Anexos

Anexo 1: Código de la función *exec* de la clase *hiloDeterminarFotogramasClaves*.

```
   void hiloDeterminarFotogramasClaves::exec() {
1      while(estado) {
2          if(!ptrListaFotogramas->isEmpty()) {
3              bloqueo.lock();
348                for(int i=0; i < ptrListaFotogramas->size(); i++) {
5                      Mat fotogramaMat = ptrListaFotogramas->at(i);

5                          IplImage *fotogramaIpl = new IplImage(fotogramaMat);
5                          IplImage *fotogramaIplGris = cvCreateImage(cvGetSize(fotogramaIpl), IPL_DEPTH_8U, 1);
5                          cvCvtColor(fotogramaIpl, fotogramaIplGris, CV_RGB2GRAY);

5                          CvHistogram *histograma;
5                          histograma = cvCreateHist(1, &celdas, CV_HIST_ARRAY);
5                          cvCalcHist(&fotogramaIplGris, histograma);

5                         int totalPixeles = fotogramaMat.rows * fotogramaMat.cols;
5                         int porcientoCanal0 = std::abs((cvGetReal1D(histograma->bins, 0) - cvGetReal1D(ultimoHistogramaClave->bins, 0))) * 100 / totalPixeles;
5                         int porcientoCanal1 = std::abs((cvGetReal1D(histograma->bins, 1) - cvGetReal1D(ultimoHistogramaClave->bins, 1))) * 100 / totalPixeles;
5                         int porcientoCanal2 = std::abs((cvGetReal1D(histograma->bins, 2) - cvGetReal1D(ultimoHistogramaClave->bins, 2))) * 100 / totalPixeles;
5                         int porcientoCanal3 = std::abs((cvGetReal1D(histograma->bins, 3) - cvGetReal1D(ultimoHistogramaClave->bins, 3))) * 100 / totalPixeles;

5                         if(tiempoAgotado || porcientoCanal0 > 10 || porcientoCanal1 > 10 || porcientoCanal2 > 10 || porcientoCanal3 > 10) {
6                             ultimoFotogramaClave = fotogramaMat;
6                             ultimoHistogramaClave = histograma;

6                             tiempoAgotado = false;
6                             disconnect(ptrReloj, SIGNAL(timeout()), this, SLOT(slotReloj()));
6                             ptrReloj->stop();
6                             connect(ptrReloj, SIGNAL(timeout()), this, SLOT(slotReloj()));
6                             ptrReloj->start(1000);

6                             ptrHiloDeterminarCalidadVideo->InsertarFotogramaClave(ultimoFotogramaClave);
                          }
7                         delete fotogramaIpl;
7                         cvReleaseImage(&fotogramaIplGris);
7                         cvReleaseHist(&histograma);
                  }
9                   ptrListaFotogramas->clear();
9              bloqueo.unlock();
           }
       }
10 }
```

Fig 11: Código de la función *exec* de la clase *hiloDeterminarFotogramasClaves*.

Anexo 2: Código de la función *DeterminarBrillo* de la clase *hiloDeterminarCalidadVideo*.

```
   void hiloDeterminarCalidadVideo::DeterminarBrillo(int pTamanoFotograma, IplImage *pFotogramaIplGris) {
1      CvHistogram *histograma;
1      int celdas = 16;
1      histograma = cvCreateHist(1, &celdas, CV_HIST_ARRAY);
1      cvCalcHist(&pFotogramaIplGris, histograma);

1      int cantidadPixelesOscuros   = 0;
1      int cantidadPixelesLuminosos = 0;

1      int i=0;
2      do {
2          cantidadPixelesOscuros += cvGetReal1D(histograma->bins, i);
2          i++;
       }
3      while(i<5);

4      i=15;
5      do {
5          cantidadPixelesLuminosos += cvGetReal1D(histograma->bins, i);
5          i--;
       }
6      while(i>9);

7      int porcientoPixelesOscuros = cantidadPixelesOscuros   * 100 / pTamanoFotograma;
7      int porcientoPixelesLuminosos  = cantidadPixelesLuminosos * 100 / pTamanoFotograma;

7      QVariant evaluacion = "Brillo. ";
7      if(porcientoPixelesOscuros >= 70) {
8          QString porcientoPixelesOscurosQString = QString::number(porcientoPixelesOscuros);
8          evaluacion = evaluacion.toString() + "Oscuridad: " + porcientoPixelesOscurosQString + "%";
8          listaEvaluaciones.append(evaluacion);
       }
9      else if(porcientoPixelesLuminosos >= 70) {
10         QString porcientoPixelesLuminososQString = QString::number(porcientoPixelesLuminosos);
10         evaluacion = evaluacion.toString() + "Luminosidad: " + porcientoPixelesLuminososQString + "%";
10         listaEvaluaciones.append(evaluacion);
       }
11     else {
11         evaluacion = evaluacion.toString() + "Buena distribución";
11         listaEvaluaciones.append(evaluacion);
       }
12     cvReleaseHist(&histograma);
12 }
```

Fig 12: Código de la función *DeterminarBrillo* de la clase *hiloDeterminarCalidadVideo*.

Anexo 3: Código de la función *DeterminarColor* de la clase *hiloDeterminarCalidadVideo*.

```
   void hiloDeterminarCalidadVideo::DeterminarColor(int pTamanoFotograma, IplImage *pFotogramaIplGris) {
1      CvHistogram *histograma;
1      int celdas = 8;
1      histograma = cvCreateHist(1, &celdas, CV_HIST_ARRAY);
1      cvCalcHist(&pFotogramaIplGris, histograma);

1      QVariant evaluacion = "Color. ";
1      bool colorPredominante = false;

1      int i=0;
2      do {
2          int porcientoPixelesMismoColor = cvGetReal1D(histograma->bins, i) * 100 / pTamanoFotograma;
2          i++;

2          if(porcientoPixelesMismoColor >= 50) {
3              QString porcientoPixelesMismoColorQString = QString::number(porcientoPixelesMismoColor);
3              evaluacion = evaluacion.toString() + "Predominio de un sólo color: " + porcientoPixelesMismoColorQString + "%";
3              listaEvaluaciones.append(evaluacion);
3              colorPredominante = true;
           }
       }
4      while(!colorPredominante && i<8);

5      if(!colorPredominante) {
6          evaluacion = evaluacion.toString() + "Buena distribución";
6          listaEvaluaciones.append(evaluacion);
       }
7      cvReleaseHist(&histograma);
7  }
```

Fig 13: Código de la función *DeterminarColor* de la clase *hiloDeterminarCalidadVideo*.

Anexo 4: Código de la función *DeterminarSaturacion* de la clase *hiloDeterminarCalidadVideo*.

```
    void hiloDeterminarCalidadVideo::DeterminarSaturacion(int pTamanoFotograma, Mat pFotogramaMat) {
  1     Mat fotogramaHSV;
  1     cvtColor(pFotogramaMat, fotogramaHSV, CV_RGB2HSV);

  1     int cantidadPixelesSaturados = 0;
  1     int nivelSaturacion;

127     for(int i=0; i < fotogramaHSV.rows; i++) {
348         for(int j=0; j < fotogramaHSV.cols; j++) {
  5             nivelSaturacion = fotogramaHSV.at<Vec3b>(i,j)[1];
  5             if(nivelSaturacion > 146)
  6                 cantidadPixelesSaturados++;
            }
        }

  9     QVariant evaluacion = "Saturación. ";
  9     int porcientoImagenSaturada = cantidadPixelesSaturados * 100 / pTamanoFotograma;

  9     if(porcientoImagenSaturada >= 70) {
 10         QString porcientoImagenSaturadaQString = QString::number(porcientoImagenSaturada);
 10         evaluacion = evaluacion.toString() + "Alta saturación: " + porcientoImagenSaturadaQString + "%";
 10         listaEvaluaciones.append(evaluacion);
        }
 11     else if(porcientoImagenSaturada <= 10) {
 12         QString porcientoImagenSaturadaQString = QString::number(porcientoImagenSaturada);
 12         evaluacion = evaluacion.toString() + "Baja saturación: " + porcientoImagenSaturadaQString + "%";
 12         listaEvaluaciones.append(evaluacion);
        }
 13     else {
 13         evaluacion = evaluacion.toString() + "Buena saturación";
 13         listaEvaluaciones.append(evaluacion);
        }
 14     cvReleaseHist(&histograma);
 14 }
```

Fig 14: Código de la función *DeterminarSaturacion* de la clase *hiloDeterminarCalidadVideo*.

Anexo 5: Código de la función *DeterminarContraste* de la clase *hiloDeterminarCalidadVideo*.

```
void hiloDeterminarCalidadVideo::DeterminarContraste(IplImage *pFotogramaIplGris) {
    CvHistogram *histograma;
    int celdas = 256;
    histograma = cvCreateHist(1, &celdas, CV_HIST_ARRAY);
    cvCalcHist(&pFotogramaIplGris, histograma);

    int cantidadNivelesGrisPresentes = 0;

    int i=0;
    do {
        if(cvGetReal1D(histograma->bins, i) >= 300)
            cantidadNivelesGrisPresentes++;
        i++;
    }
    while(i<256);

    int porcientoNivelesGrisPresentes = cantidadNivelesGrisPresentes * 100 / 256;
    QVariant evaluacion = "Contraste. ";

    if(porcientoNivelesGrisPresentes >= 40) {
        QString porcientoNivelesGrisPresentesQString = QString::number(porcientoNivelesGrisPresentes);
        evaluacion = evaluacion.toString() + "Buen contraste: " + porcientoNivelesGrisPresentesQString + "%";
        listaEvaluaciones.append(evaluacion);
    }
    else {
        QString porcientoNivelesGrisPresentesQString = QString::number(porcientoNivelesGrisPresentes);
        evaluacion = evaluacion.toString() + "Poco contraste: " + porcientoNivelesGrisPresentesQString + "%";
        listaEvaluaciones.append(evaluacion);
    }
    cvReleaseHist(&histograma);
}
```

Fig 15: Código de la función *DeterminarContraste* de la clase *hiloDeterminarCalidadVideo*.